

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

IN RE APPLICATION OF: Tatsunori KANAI, et al.

GAU:

SERIAL NO: NEW APPLICATION

EXAMINER:

FILED: HERewith

FOR: INFORMATION PROCESSING SYSTEM INCLUDING PROCESSORS AND MEMORY MANAGING
METHOD USED IN THE SAME SYSTEM

REQUEST FOR PRIORITY

COMMISSIONER FOR PATENTS
ALEXANDRIA, VIRGINIA 22313

SIR:

- ☐ Full benefit of the filing date of U.S. Application Serial Number , filed , is claimed pursuant to the provisions of 35 U.S.C. §120.
- ☐ Full benefit of the filing date(s) of U.S. Provisional Application(s) is claimed pursuant to the provisions of 35 U.S.C. §119(e): Application No. Date Filed
- ☒ Applicants claim any right to priority from any earlier filed applications to which they may be entitled pursuant to the provisions of 35 U.S.C. §119, as noted below.

In the matter of the above-identified application for patent, notice is hereby given that the applicants claim as priority:

<u>COUNTRY</u>	<u>APPLICATION NUMBER</u>	<u>MONTH/DAY/YEAR</u>
Japan	2003-185416	June 27, 2003

Certified copies of the corresponding Convention Application(s)

- ☒ are submitted herewith
- ☐ will be submitted prior to payment of the Final Fee
- ☐ were filed in prior application Serial No. filed
- ☐ were submitted to the International Bureau in PCT Application Number
Receipt of the certified copies by the International Bureau in a timely manner under PCT Rule 17.1(a) has been acknowledged as evidenced by the attached PCT/IB/304.
- ☐ (A) Application Serial No.(s) were filed in prior application Serial No. filed ; and
- ☐ (B) Application Serial No.(s)
- ☐ are submitted herewith
- ☐ will be submitted prior to payment of the Final Fee

Respectfully Submitted,

OBLON, SPIVAK, McCLELLAND,
MAIER & NEUSTADT, P.C.


Marvin J. Spivak

Registration No. 24,913

C. Irvin McClelland
Registration Number 21,124

Customer Number

22850

Tel. (703) 413-3000
Fax. (703) 413-2220
(OSMMN 05/03)

日 本 国 特 許 庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日
Date of Application: 2 0 0 3 年 6 月 2 7 日

出 願 番 号
Application Number: 特 願 2 0 0 3 - 1 8 5 4 1 6
[ST. 10/C]: [J P 2 0 0 3 - 1 8 5 4 1 6]

出 願 人
Applicant(s): 株式会社東芝

2 0 0 3 年 1 0 月 7 日

特許庁長官
Commissioner,
Japan Patent Office

今 井 康



【書類名】 特許願

【整理番号】 A000303289

【提出日】 平成15年 6月27日

【あて先】 特許庁長官 殿

【国際特許分類】 G06F 15/00

【発明の名称】 情報処理システムおよびメモリ管理方法

【請求項の数】 12

【発明者】

【住所又は居所】 神奈川県川崎市幸区小向東芝町 1 番地 株式会社東芝研
究開発センター内

【氏名】 金井 達徳

【発明者】

【住所又は居所】 神奈川県川崎市幸区小向東芝町 1 番地 株式会社東芝研
究開発センター内

【氏名】 前田 誠司

【発明者】

【住所又は居所】 神奈川県川崎市幸区小向東芝町 1 番地 株式会社東芝研
究開発センター内

【氏名】 吉井 謙一郎

【特許出願人】

【識別番号】 000003078

【氏名又は名称】 株式会社 東芝

【代理人】

【識別番号】 100058479

【弁理士】

【氏名又は名称】 鈴江 武彦

【電話番号】 03-3502-3181

【選任した代理人】

【識別番号】 100091351

【弁理士】

【氏名又は名称】 河野 哲

【選任した代理人】

【識別番号】 100088683

【弁理士】

【氏名又は名称】 中村 誠

【選任した代理人】

【識別番号】 100108855

【弁理士】

【氏名又は名称】 蔵田 昌俊

【選任した代理人】

【識別番号】 100084618

【弁理士】

【氏名又は名称】 村松 貞男

【選任した代理人】

【識別番号】 100092196

【弁理士】

【氏名又は名称】 橋本 良郎

【手数料の表示】

【予納台帳番号】 011567

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 情報処理システムおよびメモリ管理方法

【特許請求の範囲】

【請求項 1】 第 1 のローカルメモリを有する第 1 のプロセッサと、
第 2 のローカルメモリを有する第 2 のプロセッサと、
第 3 のローカルメモリを有する第 3 のプロセッサと、

前記第 1 のプロセッサによって実行される第 1 のスレッドの実効アドレス空間の一部に、前記第 1 のスレッドとの相互作用を行う第 2 のスレッドが実行される前記第 2 および第 3 の一方のプロセッサに対応する前記第 2 および第 3 の一方のローカルメモリをマッピングする手段と、

前記第 2 のスレッドが実行されるプロセッサが前記第 2 および第 3 の一方のプロセッサから他方のプロセッサに変更された場合、前記第 1 のスレッドの実効アドレス空間の一部にマッピングされるローカルメモリを前記第 2 および第 3 の一方のローカルメモリから他方のローカルメモリに変更する手段とを具備することを特徴とする情報処理システム。

【請求項 2】 前記第 1 のプロセッサ、前記第 2 のプロセッサ、および前記第 3 のプロセッサによって共有される共有メモリと、

前記第 2 のスレッドの実行が停止される場合、前記第 2 のスレッドが実行されていた前記第 2 および第 3 の一方のプロセッサに対応する前記第 2 および第 3 の一方のローカルメモリの内容を前記共有メモリ上のメモリ領域に保存する手段と

、
前記第 1 のスレッドの実効アドレス空間の一部にマッピングされるローカルメモリを前記第 2 および第 3 の一方のローカルメモリから前記共有メモリ上のメモリ領域に変更する手段とをさらに具備することを特徴とする請求項 1 記載の情報処理システム。

【請求項 3】 前記第 2 のスレッドが前記第 2 および第 3 の一方のプロセッサによって実行再開される場合、前記共有メモリ上の前記メモリ領域の内容を前記第 2 および第 3 の一方のプロセッサに対応する前記第 2 および第 3 の一方のローカルメモリに復元する手段と、

前記第 1 のスレッドの実効アドレス空間の一部にマッピングされるローカルメモリを前記共有メモリ上の前記メモリ領域から前記第 2 および第 3 の一方のローカルメモリに変更する手段とをさらに具備することを特徴とする請求項 2 記載の情報処理システム。

【請求項 4】 前記第 2 のスレッドが前記第 2 および第 3 の他方のプロセッサによって実行再開される場合、前記共有メモリ上の前記メモリ領域の内容を前記第 2 および第 3 の他方のプロセッサに対応する前記第 2 および第 3 の他方のローカルメモリに復元する手段と、

前記第 1 のスレッドの実効アドレス空間の一部にマッピングされるローカルメモリを前記共有メモリ上の前記メモリ領域から前記第 2 および第 3 の他方のローカルメモリに変更する手段とをさらに具備することを特徴とする請求項 2 記載の情報処理システム。

【請求項 5】 第 1 のローカルメモリを有する第 1 のプロセッサ、第 2 のローカルメモリを有する第 2 のプロセッサ、および第 3 のローカルメモリを有する第 3 のプロセッサを含む情報処理システムにおいてスレッド間の通信に用いられるローカルメモリを管理するメモリ管理方法であって、

前記第 1 のプロセッサによって実行される第 1 のスレッドの実効アドレス空間の一部に、前記第 1 のスレッドとの相互作用を行う第 2 のスレッドが実行される前記第 2 および第 3 の一方のプロセッサに対応する前記第 2 および第 3 の一方のローカルメモリをマッピングするステップと、

前記第 2 のスレッドが実行されるプロセッサが前記第 2 および第 3 の一方のプロセッサから他方のプロセッサに変更された場合、前記第 1 のスレッドの実効アドレス空間の一部にマッピングされるローカルメモリを前記第 2 および第 3 の一方のローカルメモリから他方のローカルメモリに変更するステップとを具備することを特徴とするメモリ管理方法。

【請求項 6】 前記第 2 のスレッドの実行が停止される場合、前記第 2 のスレッドが実行されていた前記第 2 および第 3 の一方のプロセッサに対応する前記第 2 および第 3 の一方のローカルメモリの内容を、前記第 1 のプロセッサ、前記第 2 のプロセッサ、および前記第 3 のプロセッサによって共有される共有メモリ

上のメモリ領域に保存するステップと、

前記第1のスレッドの実効アドレス空間の一部にマッピングされるローカルメモリを前記第2および第3の一方のローカルメモリから前記共有メモリ上のメモリ領域に変更するステップとをさらに具備することを特徴とする請求項5記載のメモリ管理方法。

【請求項7】 前記第2のスレッドが前記第2および第3の一方のプロセッサによって実行再開される場合、前記共有メモリ上の前記メモリ領域の内容を前記第2および第3の一方のプロセッサに対応する前記第2および第3の一方のローカルメモリに復元するステップと、

前記第1のスレッドの実効アドレス空間の一部にマッピングされるローカルメモリを前記共有メモリ上の前記メモリ領域から前記第2および第3の一方のローカルメモリに変更するステップとをさらに具備することを特徴とする請求項6記載のメモリ管理方法。

【請求項8】 前記第2のスレッドが前記第2および第3の他方のプロセッサによって実行再開される場合、前記共有メモリ上の前記メモリ領域の内容を前記第2および第3の他方のプロセッサに対応する前記第2および第3の他方のローカルメモリに復元するステップと、

前記第1のスレッドの実効アドレス空間の一部にマッピングされるローカルメモリを前記共有メモリ上の前記メモリ領域から前記第2および第3の他方のローカルメモリに変更するステップとをさらに具備することを特徴とする請求項6記載のメモリ管理方法。

【請求項9】 第1のローカルメモリを有する第1のプロセッサ、第2のローカルメモリを有する第2のプロセッサ、および第3のローカルメモリを有する第3のプロセッサを含むコンピュータに、スレッド間の通信に用いられるローカルメモリを管理させるプログラムであって、

前記第1のプロセッサによって実行される第1のスレッドの実効アドレス空間の一部に、前記第1のスレッドとの相互作用を行う第2のスレッドが実行される前記第2および第3の一方のプロセッサに対応する前記第2および第3の一方のローカルメモリをマッピングする処理を、前記コンピュータに実行させる手順と

前記第 2 のスレッドが実行されるプロセッサが前記第 2 および第 3 の一方のプロセッサから他方のプロセッサに変更された場合、前記第 1 のスレッドの実効アドレス空間の一部にマッピングされるローカルメモリを前記第 2 および第 3 の一方のローカルメモリから他方のローカルメモリに変更する処理を、前記コンピュータに実行させる手順とを具備することを特徴とするプログラム。

【請求項 10】 前記第 2 のスレッドの実行が停止される場合、前記第 2 のスレッドが実行されていた前記第 2 および第 3 の一方のプロセッサに対応する前記第 2 および第 3 の一方のローカルメモリの内容を、前記第 1 のプロセッサ、前記第 2 のプロセッサ、および前記第 3 のプロセッサによって共有される共有メモリ上のメモリ領域に保存する処理を、前記コンピュータに実行させる手順と、

前記第 1 のスレッドの実効アドレス空間の一部にマッピングされるローカルメモリを前記第 2 および第 3 の一方のローカルメモリから前記共有メモリ上のメモリ領域に変更する処理を、前記コンピュータに実行させる手順とをさらに具備することを特徴とする請求項 9 記載のプログラム。

【請求項 11】 前記第 2 のスレッドが前記第 2 および第 3 の一方のプロセッサによって実行再開される場合、前記共有メモリ上の前記メモリ領域の内容を前記第 2 および第 3 の一方のプロセッサに対応する前記第 2 および第 3 の一方のローカルメモリに復元する処理を、前記コンピュータに実行させる手順と、

前記第 1 のスレッドの実効アドレス空間の一部にマッピングされるローカルメモリを前記共有メモリ上の前記メモリ領域から前記第 2 および第 3 の一方のローカルメモリに変更する処理を、前記コンピュータに実行させる手順とをさらに具備することを特徴とする請求項 10 記載のプログラム。

【請求項 12】 前記第 2 のスレッドが前記第 2 および第 3 の他方のプロセッサによって実行再開される場合、前記共有メモリ上の前記メモリ領域の内容を前記第 2 および第 3 の他方のプロセッサに対応する前記第 2 および第 3 の他方のローカルメモリに復元する処理を、前記コンピュータに実行させる手順と、

前記第 1 のスレッドの実効アドレス空間の一部にマッピングされるローカルメモリを前記共有メモリ上の前記メモリ領域から前記第 2 および第 3 の他方のロー

カルメモリに変更する処理を、前記コンピュータに実行させる手順とをさらに具備することを特徴とする請求項10記載のプログラム。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】

本発明は複数のプロセッサを含む情報処理システムおよび同システムで用いられるメモリ管理方法に関する。

【0002】

【従来の技術】

従来より、サーバコンピュータのような計算機システムにおいては、その演算処理能力の向上を図るために、マルチプロセッサ、並列プロセッサのようなシステムアーキテクチャが利用されている。マルチプロセッサおよび並列プロセッサのどちらも、複数のプロセッサユニットを利用することによって演算処理の並列化を実現している。

複数のプロセッサユニットを備えたシステムとしては、例えば、1台の高速CPU、複数台の低速CPU、および共有メモリを備えたシステムが知られている（例えば、特許文献1参照）。このシステムにおいては、高速CPUおよび複数台の低速CPUに対する処理プログラムのプロセス群の割付は、プロセス群の並列動作度の大小および処理時間の大小に応じて行われる。

【0003】

ところで、最近では、計算機システムのみならず、例えば、AV（オーディオ・ビデオ）データのような大容量のデータをリアルタイムに処理する組み込み機器においても、その演算処理能力の向上のためにマルチプロセッサ、並列プロセッサのようなシステムアーキテクチャの導入が要求されている。

【0004】

【特許文献1】

特開平10-143380号公報

【0005】

【発明が解決しようとする課題】

しかし、複数のプロセッサを含むシステムアーキテクチャを前提としたリアルタイム処理システムの報告はほとんどなされていないのが現状である。

リアルタイム処理システムにおいては、ある許容時間時間の制限内に個々の処理を完了することが要求される。しかし、マルチプロセッサ、並列プロセッサのようなシステムアーキテクチャをリアルタイム処理システムに適用した場合においては、共有メモリに対するアクセスの競合、メモリバスのバンド幅の制約などにより、複数のプロセッサそれぞれの性能を十分に活用できなくなるという問題が生じる。また互いに異なるプロセッサによって実行されるスレッド間でデータを受け渡すための通信も共有メモリ上のバッファを介して行われるので、頻繁に相互作用を行うスレッド間においてはその通信に関するレイテンシが大きな問題となる。

【0006】

本発明は上述の事情を考慮してなされたものであり、複数のプロセッサを用いて複数のスレッドを効率よく並列に実行することが可能な情報処理システムおよびメモリ管理方法を提供することを目的とする。

【0007】

【課題を解決するための手段】

上述の課題を解決するため、本発明の情報処理システムは、第1のローカルメモリを有する第1のプロセッサと、第2のローカルメモリを有する第2のプロセッサと、第3のローカルメモリを有する第3のプロセッサと、前記第1のプロセッサによって実行される第1のスレッドの実効アドレス空間の一部に、前記第1のスレッドとの相互作用を行う第2のスレッドが実行される前記第2および第3の一方のプロセッサに対応する前記第2および第3の一方のローカルメモリをマッピングする手段と、前記第2のスレッドが実行されるプロセッサが前記第2および第3の一方のプロセッサから他方のプロセッサに変更された場合、前記第1のスレッドの実効アドレス空間の一部にマッピングされるローカルメモリを前記第2および第3の一方のローカルメモリから他方のローカルメモリに変更する手段とを具備することを特徴とする。

【0008】

この情報処理システムにおいては、各プロセッサにローカルメモリが設けられているので、各スレッドは共有メモリをアクセスせずとも、プロセッサ内のローカルメモリをアクセスするだけでプログラムを実行することができる。また、相互作用を行う相手のスレッドが実行されるプロセッサに応じて、実効アドレス空間にマッピングされる、相手のスレッドに対応するプロセッサのローカルメモリが自動的に変更されるので、各スレッドは相手のスレッドがディスパッチされるプロセッサを意識することなく、相手のスレッドとの相互作用を効率よく行うことが出来る。よって、複数のスレッドを効率よく並列に実行することが可能となる。

【0009】

【発明の実施の形態】

以下、図面を参照して本発明の実施形態を説明する。

図1には、本発明の一実施形態に係るリアルタイム処理システムを実現するための計算機システムの構成例が示されている。この計算機システムは、リアルタイム性が要求される各種処理をその時間的な制約条件の範囲内で実行する情報処理システムであり、汎用計算機として利用できるほか、リアルタイム性が要求される処理を実行するための様々な電子機器用の埋め込みシステムとして利用することができる。図1に示されているように、この計算機システムにおいては、マスタープロセッサユニット(MPU11: Master Processing Unit)11と、複数のバーサタイルプロセッサユニット(VPU: Versatile Processing Unit)12と、メインメモリ14と、入出力制御装置15とが、接続装置13によって相互に接続されている。接続装置13は、例えば、クロスバススイッチのような相互結合網、あるいはバスによって構成されている。リング状のバス構造を用いることも出来る。MPU11は計算機システムの動作を制御するメインプロセッサである。オペレーティングシステム(OS: Operating System)は、主にMPU11によって実行される。OSの一部の機能はVPU12や入出力制御装置15で分担して実行することもできる。各VPU12は、MPU11の管理の下で各種の処理を実行するプロセッサである。MPU11は、複数のVPU12に処理を振り分けて並列に実行させるための制御を行う。これにより高速で効率よい処理

の実行を行うことが出来る。メインメモリ 14 は、MPU 11、複数の VPU 12 および入出力制御装置 15 によって共有される記憶装置（共有メモリ）である。OS およびアプリケーションプログラムはメインメモリ 14 に格納される。入出力制御装置 15 には、ひとつあるいは複数の入出力デバイス（入出力装置）16 が接続される。入出力制御装置 15 はブリッジとも呼ばれる。

【0010】

接続装置 15 はデータ転送レートを保証するQoS機能を持つ。この機能は、接続装置 15 を介したデータ転送を予約されたバンド幅（転送速度）で実行することによって実現される。QoS機能は、たとえば、ある VPU 12 からメモリ 14 に 5Mbps でライトデータを送信する場合、あるいはある VPU 12 と別の VPU 12 との間で 100Mbps でデータ転送する場合に利用される。VPU 12 は接続装置 13 に対してバンド幅（転送速度）を指定（予約）する。接続装置 13 は指定されたバンド幅を要求した VPU 12 に対して優先的に割り当てる。ある VPU 12 のデータ転送に対してバンド幅が予約されたならば、その VPU 12 によるデータ転送中に他の VPU 12、MPU 11 あるいは入出力制御装置 15 が大量のデータ転送を行っても、予約されたバンド幅は確保される。この機能は、特に、リアルタイム処理を行う計算機にとって重要な機能である。

【0011】

図1の構成では、MPU 11 が1つ、VPU 12 が4つ、メモリ 14 が1つ、入出力制御装置が1つであるが、VPU 12 の個数は制限されない。また MPU 11 を持たない構成も可能である。この場合、MPU 11 の行う処理は、ある一つの VPU 12 が担当する。つまり、仮想的な MPU 11 の役割を VPU が兼ねる。

【0012】

図2には、MPU 11 と各 VPU 12 の構成が示されている。MPU 11 は処理ユニット 21 およびメモリ管理ユニット 22 を備えている。処理ユニット 21 は、メモリ管理ユニット 22 を通してメモリ 14 をアクセスする。メモリ管理ユニット 22 は、仮想記憶管理と、メモリ管理ユニット 22 内のキャッシュメモリの管理を行うユニットである。各 VPU 12 は、処理ユニット 31、ローカルス

トレージ（ローカルメモリ）32、およびメモリコントローラ33を備えている。各VP U 12の処理ユニット31は、そのVP U内部のローカルストレージ32を直接アクセスすることができる。メモリコントローラ33は、ローカルストレージ32とメモリ14の間のデータ転送を行うDMAコントローラの役割を持つ。このメモリコントローラ33は、接続装置14のQoS機能を利用できるように構成されており、バンド幅を予約する機能および予約したバンド幅でデータ入出力を行う機能を有している。またメモリコントローラ33は、MP U 11のメモリ管理ユニット22と同様の仮想記憶管理機能を持つ。VP U 12の処理ユニット31はローカルストレージ32を主記憶として使用する。処理ユニット31はメモリ14に対して直接的にアクセスするのではなく、メモリコントローラ33に指示して、メモリ14の内容をローカルストレージ32に転送して読んだり、ローカルストレージ32の内容をメモリ14に書いたりする。

【0013】

MP U 11のメモリ管理ユニット22およびVP U 12のメモリコントローラ33それぞれによって実行される仮想記憶管理は、たとえば図3のように実施することができる。MP U 11の処理ユニット21あるいはVP U 12のメモリコントローラ33から見たアドレスは、図3の上の部分に示すような64ビットのアドレスである。この64ビットのアドレスは、上位の36ビットがセグメント番号、中央の16ビットがページ番号、下位の12ビットがページオフセットである。このアドレスから、実際に接続装置13を通してアクセスする実アドレス空間への変換は、セグメントテーブル50およびページテーブル60を用いて実行される。セグメントテーブル50およびページテーブル60は、メモリ管理ユニット22およびメモリコントローラ33に各々設けられている。

【0014】

MP U 11および各VP U 12から見た実アドレス（RA）空間には、図4に示すように、たとえば以下のようなデータがマッピングされている。

1. メモリ（主記憶装置）
2. MP U 11の各種制御レジスタ
3. 各VP U 12の各種制御レジスタ

4. 各 V P U 1 2 のローカルストレージ

5. 各種入出力デバイス（入出力装置）の制御レジスタ（入出力制御装置の制御レジスタも含む）

M P U 1 1 および各 V P U 1 2 は、実アドレス空間の該当するアドレスにアクセスすることで、1～5の各データを読み書きすることができる。特に、実アドレス空間にアクセスすることで、どの M P U 1 1 からでも、あるいはどの V P U 1 2 からでも、さらに入出力制御装置 1 5 からでも、任意の V P U 1 2 のローカルストレージ 3 2 にアクセスすることができることは重要である。またセグメントテーブルあるいはページテーブルを用いて、V P U 1 2 のローカルストレージ 3 2 の内容が自由に読み書きされないように保護することもできる。

M P U 1 1 あるいは V P U 1 2 からみたアドレス空間は、図 3 の仮想記憶メカニズムを用いて、たとえば図 5 に示すようにマッピングされる。M P U 1 1 あるいは V P U 1 2 上で実行しているプログラムから直接見えるのは、実効アドレス（E A ; Effective Address）空間である。E A は、セグメントテーブル 5 0 によって、仮想アドレス（V A ; Virtual Address）空間にマッピングされる。さらに V A は、ページテーブル 6 0 によって、実アドレス（R A ; Real Address）空間にマップされる。この R A が、図 4 で説明したような構造を持っている。

【0015】

M P U 1 1 は制御レジスタ等のハードウェア機構によって、例えば、各 V P U 1 2 のレジスタの読み書き、各 V P U 1 2 のプログラムの実行開始／停止などの、各 V P U 1 2 の管理を行うことができる。また、M P U 1 1 と V P U 1 2 の間、あるいはある V P U 1 2 と他の V P U 1 2 の間の通信や同期は、メールボックスやイベントフラグなどのハードウェア機構によって行うことができる。

【0016】

この実施形態の計算機システムは、従来ハードウェアで実現されていたようなリアルタイム性の要求の厳しい機器の動作を、ソフトウェアを用いて実現することを可能にする。例えば、ある V P U 1 2 があるハードウェアを構成するある幾つかのハードウェアコンポーネントに対応する演算処理を実行し、それと並行して、他の V P U 1 2 が他の幾つかのハードウェアコンポーネントに対応する演算

処理を実行する。

【0017】

図6はデジタルテレビ放送の受信機の簡略化したハードウェア構成を示している。図6においては、受信した放送信号はDEMUX（デマルチプレクサ）回路101によって音声データと映像データと字幕データそれぞれに対応する圧縮符号化されたデータストリームに分解される。圧縮符号化された音声データストリームはA-DEC（音声デコーダ）回路102によってデコードされる。圧縮符号化された映像データストリームはV-DEC（映像デコーダ）回路103によってデコードされる。デコードされた映像データストリームはPROG（プログレッシブ変換）回路105に送られ、そこでプログレッシブ映像信号に変換するためのプログレッシブ変換処理が施される。プログレッシブ変換された映像データストリームはBLEND（画像合成）回路106に送られる。字幕データストリームはTEXT（字幕処理）回路104によって字幕の映像に変換された後、BLEND回路106に送られる。BLEND回路106は、PROG回路105から送られてくる映像と、TEXT回路104から送られてくる字幕映像とを合成して、映像ストリームとして出力する。この一連の処理が、映像のフレームレート（たとえば、1秒間に30コマ、32コマ、または60コマ）に従って、繰り返し実行される。

【0018】

図6のようなハードウェアの動作をソフトウェアによって実行するために、本実施形態では、たとえば図7に示すように、図6のハードウェアの動作をソフトウェアとして実現したプログラムモジュール100を用意する。このプログラムモジュール100は、複数の処理要素の組み合わせから構成されるリアルタイム処理を計算機システムに実行させるためのアプリケーションプログラムであり、マルチスレッドプログラミングを用いて記述されている。このプログラムモジュール100は、図6のハードウェアコンポーネント群に対応する複数の処理要素それぞれに対応した手順を記述した複数のプログラム111～116を含んでいる。すなわち、プログラムモジュール100には、DEMUXプログラム111、A-DECプログラム112、V-DECプログラム113、TEXTプログ

ラム 114、PROG プログラム 115、および BLEND プログラム 116 が含まれている。DEMUX プログラム 111、A-DEC プログラム 112、V-DEC プログラム 113、TEXT プログラム 114、PROG プログラム 115、および BLEND プログラム 116 は、それぞれ図 6 の DEMUX 回路 101、A-DEC 回路 102、V-DEC 回路 103、TEXT 回路 104、PROG 回路 105、および BLEND 回路 106 に対応する処理を実行するためのプログラムであり、それぞれスレッドとして実行される。つまり、プログラムモジュール 100 の実行時には、DEMUX プログラム 111、A-DEC プログラム 112、V-DEC プログラム 113、TEXT プログラム 114、PROG プログラム 115、および BLEND プログラム 116 それぞれに対応するスレッドが生成され、生成されたスレッドそれぞれが 1 以上の VPU 12 にディスパッチされて実行される。VPU 12 のローカルストレージ 32 にはその VPU 12 にディスパッチされたスレッドに対応するプログラムがロードされ、スレッドはローカルストレージ 32 上のプログラムを実行する。デジタルテレビ放送の受信機を構成するハードウェアモジュール群それぞれに対応するプログラム 111~116 と、構成記述 117 と呼ぶデータとをパッケージ化したものが、デジタルテレビ放送の受信機を実現するプログラムモジュール 100 になる。

【0019】

構成記述 117 は、プログラムモジュール 100 内の各プログラム（スレッド）をどのように組み合わせて実行するべきかを示す情報であり、プログラム 111~116 間の入出力関係および各プログラムの処理に必要なコスト（時間）などを示す。図 8 には構成記述 117 の例が示されている。

【0020】

図 8 の構成記述 117 の例では、スレッドとして動作する各モジュール（プログラムモジュール 100 内の各プログラム）に対して、その入力につながるモジュール、その出力につながるモジュール、そのモジュールの実行に要するコスト、出力につながるモジュールそれぞれへの出力に必要なバッファサイズが記述されている。たとえば、番号③の V-DEC プログラムは、番号①の DEMUX プログラムの出力を入力とし、その出力は番号⑤の PROG プログラムに向かって

おり、その出力に必要なバッファは1MBで、番号③のV-DECプログラム自体の実行コストは50であることを示している。なお、実行に必要なコストは、実行に必要な時間（実行期間）やステップ数などを単位として記述することができる。また、何らかの仮想的な仕様のVPUで実行した場合の時間を単位とすることも可能である。計算機によってVPUの仕様や処理性能が異なる場合もあるので、このように仮想的な単位を設けてコストを表現するのは望ましい形態である。図8に示した構成記述117に従って実行する場合の、プログラム間のデータの流れは図9の通りである。

【0021】

さらに、構成記述117には、プログラム111～116それぞれに対応するスレッド間の結合属性を示す結合属性情報がスレッドパラメータとして記述されている。なお、スレッドパラメータはプログラム111～116中にコードとして直接記述することも可能である。

【0022】

次に、図10、図11を参照して、プログラム111～116が本実施形態の計算機システムによってどのように実行されるかを説明する。ここでは、VPU0とVPU1の2つのVPU12が計算機システムに設けられている構成を想定する。毎秒30フレームで映像を表示する場合の、各VPU12に対するプログラムの割り当てを時間を追って記入したのが図10である。ここでは周期1の間で1フレーム分の音声と映像を出力している。まず、VPU0でDEMUXプログラムが処理を行い、その結果の音声と映像と字幕のデータをバッファに書き込む。その後VPU1でA-DECプログラムとTEXTプログラムを順次実行し、それぞれの処理結果をバッファに書き込む。VPU0では、次にV-DECプログラムが映像データの処理を行い、結果をバッファに書き込む。VPU0では、続いてPROGプログラムが処理を行い、結果をバッファに書き込む。この時点で、VPU1でのTEXTの処理は終わっているので、最後のBLENDプログラムの実行をVPU0で行い、最終的な映像データを作成する。この処理の流れを、毎周期繰り返すように実行する。

【0023】

ここで説明したように、所望の動作を滞りなく行えるように、各VPU12上で、いつ、どのプログラムを実行するかを決める作業を、スケジューリングとよぶ。スケジューリングを行うモジュールをスケジューラとよぶ。本実施形態では、プログラムモジュール100中に含まれる上述の構成記述117に基づいてスケジューリングが行われる。

【0024】

図11は、毎秒60フレームで表示する場合の実行の様子を示している。図10と異なるのは、図10では毎秒30フレームだったので、1周期（1/30秒）で1フレーム分の処理を完了できたのに対し、図11では毎秒60フレーム処理する必要がある点である。すなわち、1周期（1/60秒）では1フレーム分の処理を完了できないので、図11では、複数（ここでは2）周期にまたがったソフトウェアパイプライン処理を行っている。たとえば周期1のはじめに入力された信号に対して、VPU0でDEMUX処理とV-DEC処理を行う。その後、周期2においてVPU1でA-DEC、TEXT、PROG、BLENDの各処理を行って最終的な映像データを出力する。周期2ではVPU0は次のフレームのDEMUXとV-DECの処理を行っている。このように、VPU0によるDEMUX、V-DECの処理と、VPU1によるA-DEC、TEXT、PROG、BLENDの処理を、2周期にまたがってパイプライン的に実行する。

【0025】

なお、図7に示したプログラムモジュール100は、本実施形態の計算機システムを組み込んだ機器内のフラッシュROMやハードディスクに予め記録しておいてもよいが、ネットワークを介して流通させるようにしてもよい。この場合、本実施形態の計算機システムによって実行される処理の内容は、ネットワークを介してダウンロードしたプログラムモジュールの種類に応じて決まる。よって、例えば本実施形態の計算機システムを組み込んだ機器に、様々な専用ハードウェアそれぞれに対応するリアルタイム処理を実行させることが出来る。例えば、新しいコンテンツの再生に必要な新しいプレーヤーソフトウェアやデコーダソフトウェアや暗号ソフトウェアなどを、本実施形態の計算機システムで実行可能なプログラムモジュールとして、コンテンツと一緒に配布することで、本実施形態の

計算機システムを搭載した機器であれば、いずれの機器でも、その能力が許す範囲内で、そのコンテンツを再生することができる。

【0026】

(オペレーティングシステム)

本計算機システムでは、システム内にOS（オペレーティングシステム）をひとつだけ実装する場合には、図12に示すように、そのOS 201がすべての実資源（たとえば、MPU 11、VPU 12、メモリ 14、入出力制御装置 15、入出力装置 16 など）を管理する。

一方、仮想計算機方式を用いて、複数のOSを同時に動作させることも可能である。この場合には、図13に示すように、まず仮想計算機OS 301を実装し、それがすべての実資源（たとえば、MPU 11、VPU 12、メモリ 14、入出力制御装置 15、入出力装置 16 など）を管理する。仮想計算機OS 301はホストOSと称されることもある。さらに仮想計算機OS 301の上に、ひとつ以上のOS（ゲストOSとも呼ぶ）を実装する。各ゲストOS 302, 303は、図14に示すように、仮想計算機OS 301によって与えられる仮想的な計算機資源から構成される計算機上で動作し、ゲストOS 302, 303の管理するアプリケーションプログラムに各種のサービスを提供する。図14の例では、ゲストOS 302は、1つのMPU 11と、2つのVPU 12と、メモリ 14とから構成される計算機上で動いていると思っており、ゲストOS 303は1つのMPU 11と、4つのVPU 12と、メモリ 14とから構成される計算機上で動いていると思っている。ゲストOS 302からみたVPU 12や、ゲストOS 303からみたVPU 12が、実際には実資源のどのVPU 12に対応しているかは、仮想計算機OS 301が管理している。ゲストOS 302, 303は、その対応を意識する必要はない。

【0027】

仮想計算機OS 301は、計算機システム全体の資源を時分割で各ゲストOS 302, 303に割り当てるように、ゲストOS 302, 303のスケジューリングを行う。例えば、ゲストOS 302がリアルタイム処理を行うものであるとする。たとえば1秒間に30回、正しいペースで処理を行いたい場合には、

各ゲストOS 302はそのパラメタを仮想計算機OS 301に設定する。仮想計算機OS 301は、1/30秒に1回、確実にそのゲストOS 301に必要なだけの処理時間が割り当てられるようにスケジューリングを行う。リアルタイム性を要求しない処理を行うゲストOSには、リアルタイム性を要求するゲストOSよりも低い優先度で、処理時間の割り当てを行うように、スケジューリングが行われる。図15は、時間軸を横にとって、ゲストOS 302とゲストOS 303が切り替わりながら動いている様子を示している。ゲストOS 302が動いている間は、MPU 11と全てのVPU 12がゲストOS 302の資源として使用され、ゲストOS 303が動いている間は、MPU 11と全てのVPU 12がゲストOS 303の資源として使用される。

【0028】

図16は別の動作モードを示している。ターゲットアプリケーションによってはVPU 12をずっと占有して利用したい場合がある。たとえば、常にデータやイベントを監視し続けることが必要なアプリケーションがこれに相当する。このようなときには、特定のVPU 12を特定のゲストOSによって占有するように、仮想計算機301のスケジューラがスケジュール管理する。図16では、VPU 4をゲストOS 301の専用資源に指定した場合の例である。仮想計算機OS 301がゲストOS 302（OS 1）とゲストOS 303（OS 2）を切り替えても、VPU 4は常にゲストOS 301（OS 1）の管理下で動作し続ける。

【0029】

さて、複数のVPU 12を用いてプログラムを動作させるために、本実施形態では、複数のVPU 12それぞれに割り当てるスレッドをスケジューリングするためのスケジューラを含む、VPU実行環境と呼ぶソフトウェアモジュールを用いる。本計算機システムにOSがひとつしか搭載されていない場合は、図17に示すようにそのOS 201にVPU実行環境401を実装する。この時、VPU実行環境401は、OS 201のカーネル内に実装することもできるし、ユーザプログラムレベルで実装することもできるし、両者に分割して協調して動作するように実装することも出来る。一方、仮想計算機OS上でひとつあるいは複数のOSを動作させる場合、VPU実行環境401を実装する方式には、次のような

方式がある。

1. 仮想計算機OS 301の中にVPU実行環境401を実装する方式(図18)
2. VPU実行環境401を仮想計算機OS 301が管理するひとつのOSとして実装する方式(図19)。図19では、仮想計算機OS 301上で動作するゲストOS 304自体がVPU実行環境401である。
3. 仮想計算機OS 301が管理する各ゲストOSに、それぞれ専用のVPU実行環境401を実装する方式(図20)。図20においては、ゲストOS 302, 303にそれぞれVPU実行環境401, 402が実装されている。VPU実行環境401, 402は、仮想計算機OS 301の提供するゲストOS間の通信機能を用いて、必要に応じて、互いに連携して動作する。
4. 仮想計算機OS 301が管理するゲストOSのうちのひとつにVPU実行環境401を実装して、VPU実行環境を持たないゲストOSは、仮想計算機OS 301の提供するゲストOS間の通信機能を用いて、VPU実行環境401を持つゲストOSのVPU実行環境401を利用する方式(図21)。

【0030】

これらの方式のメリットは以下のとおりである。

方式1のメリット

- ・仮想計算機OSの持つゲストOS(仮想計算機OSが管理する対象のOS)のスケジューリングと、VPU12のスケジューリングを一体化できるので、効率良く、きめ細かなスケジューリングができ、資源を有効利用できる。
- ・複数のゲストOS間でVPU実行環境を共有できるので、新しいゲストOSを導入する場合に新しくVPU実行環境を作らなくてもよい。

方式2のメリット

- ・仮想計算機OSの上にあるゲストOS間でVPU12のスケジューラを共有できるので、効率良く、きめ細かなスケジューリングができ、資源を有効利用できる。
- ・複数のゲストOS間でVPU実行環境を共有できるので、新しいゲストを導入する場合に新しくVPU実行環境を作らなくてもよい。

・ V P U 実行環境を仮想計算機 O S や特定のゲスト O S に依存せずに作れるので、標準化がしやすく、取り替えて使うことも出来る。特定の組み込み機器に適応した V P U 実行環境を作って、その機器の特性を活かしたスケジューリング等を行うことで、効率良い実行ができる。

方式 3 のメリット

・ 各ゲスト O S に対して V P U 実行環境を最適に実装できるので、効率良く、きめ細かなスケジューリングができ、資源を有効利用できる。

【 0 0 3 1 】

方式 4 のメリット

・ すべてのゲスト O S が V P U 実行環境を実装する必要がないので、新しいゲスト O S を追加しやすい。

このように、いずれの方式でも V P U 実行環境を実装することができる。また、このほかにも適宜実施可能である。

【 0 0 3 2 】

(サービスプロバイダ)

本実施形態の計算機システムにおいては、V P U 実行環境 4 0 1 は、各 V P U 1 2 に関連する各種資源（各 V P U の処理時間、メモリ、接続装置のバンド幅、など）の管理とスケジューリング機能の他に、さまざまなサービス（ネットワークを使った通信機能、ファイルの入出力機能、コーデックなどのライブラリ機能の呼び出し、ユーザとのインタフェース処理、入出力デバイスを使った入出力処理、日付や時間の読み出し、など）を提供する。これらのサービスは、V P U 1 2 上で動作するアプリケーションプログラムから呼び出されて、簡単なサービスの場合にはその V P U 1 2 上のサービスプログラムで処理される。しかし通信やファイルの処理など V P U 1 2 だけでは処理できないサービスに関しては、M P U 1 1 上のサービスプログラムによって処理する。このようなサービスを提供するプログラムを、サービスプロバイダ（S P）と呼ぶ。

【 0 0 3 3 】

図 2 2 に V P U 実行環境のひとつの実施例を示す。V P U 実行環境の主要部分は M P U 1 1 上に存在する。これが、M P U 側 V P U 実行環境 5 0 1 である。各

VPU12上には、そのVPU12内で処理可能なサービスを実行する最小限の機能のみを持つVPU側VPU実行環境502が存在する。MPU側VPU実行環境501の機能は、大きく、VPUコントロール511と、サービスブローカ512の2つに分けられる。VPUコントロール512は、主に、各VPU12に関連する各種資源（VPUの処理時間、メモリ、仮想空間、接続装置のバンド幅、など）の管理機構や、同期機構や、セキュリティの管理機構や、スケジューリング機能を提供する。スケジューリング結果に基づいてVPU12上のプログラムのディスパッチを行うのは、このVPUコントロール511である。サービスブローカ512は、VPU12上のアプリケーションが呼び出したサービス要求を受けて、適当なサービスプログラム（サービスプロバイダ）を呼び出してそのサービスを提供する。

VPU側VPU実行環境502は、主に、VPU12上のアプリケーションプログラムが呼び出したサービス要求を受けて、VPU12内で処理できるものは処理し、そうでないものはMPU側VPU実行環境501のサービスブローカ512に処理を依頼する働きをする。

【0034】

図23に、VPU側VPU実行環境502がサービス要求を処理する手順を示す。VPU側VPU実行環境502はアプリケーションプログラムからのサービス呼び出しを受け取ると（ステップS101）、VPU実行環境502内で処理できるサービスであるかどうかを判別し（ステップS102）、それであれば、対応するサービスを実行して、結果を呼び出し元へ返す（ステップS103, S107）。一方、VPU実行環境502内で処理できるサービスではないならば、該当するサービスを実行可能なサービスプログラムがVPU12上で実行可能なプログラムとして登録されているかどうかを判断する（ステップS104）。登録されているならば、当該サービスプログラムを実行し、結果を呼び出し元へ返す（ステップS105, S107）。登録されていないならば、サービスブローカ512に処理を依頼し、そしてサービスブローカ512から返されるサービスの結果を呼び出し元へ返す（ステップS106, S107）。

【0035】

図 24 に、MPU 側 VPU 実行環境 501 のサービスブローカ 512 が、VPU 側 VPU 実行環境 502 から要求されたサービス进行处理する手順を示す。サービスブローカ 512 は VPU 側 VPU 実行環境 502 からのサービス呼び出しを受け取ると（ステップ S111）、VPU 実行環境 501 内で処理できるサービスであるかどうかを判別し（ステップ S112）、それであれば、対応するサービスを実行して、結果を呼び出し元の VPU 側 VPU 実行環境 502 へ返す（ステップ S113、S114）。一方、VPU 実行環境 501 内で処理できるサービスではないならば、該当するサービスを実行可能なサービスプログラムが MPU 11 上で実行可能なプログラムとして登録されているかどうかを判断する（ステップ S114）。登録されているならば、当該サービスプログラムを実行し、結果を呼び出し元の VPU 側 VPU 実行環境 502 へ返す（ステップ S116、S114）。登録されていないならば、エラーを呼び出し元の VPU 側 VPU 実行環境 502 へ返す（ステップ S117）。

【0036】

なお、VPU 12 で実行するプログラムが発行するサービス要求には、サービスの実行結果のリプライを返すものもあれば、要求を出すだけでリプライの無いものもある。また、リプライ先は、通常は要求を出したスレッドであるが、リプライ先として他のスレッド、スレッドグループ、あるいはプロセスを指定することもできる。そのため、サービス要求のメッセージには、リプライ先の指定も含めることが好ましい。サービスブローカ 512 は、広く使われているオブジェクトリクエストブローカを用いて実現することができる。

【0037】

（リアルタイム処理）

本実施形態の計算機システムはリアルタイム処理システムとして機能する。この場合、そのリアルタイム処理システムの対象とする処理は、大きく、

1. ハードリアルタイム処理
2. ソフトリアルタイム処理
3. ベストエフォート処理（ノンリアルタイム処理）

の 3 種類に分類できる。1 と 2 がいわゆるリアルタイム処理と呼ばれるものであ

る。本実施形態のリアルタイム処理システムは、多くの既存のOSと同様、スレッドとプロセスの概念を持っている。ここではまず、本実施形態のリアルタイム処理システムにおけるスレッドとプロセスに関して説明する。

【0038】

スレッドには、次の3つのクラスがある。

1. ハードリアルタイムクラス

このスレッドクラスは、その時間要件 (timing requirements) は非常に重要で、その要件が満たされなかった際に重大な状況になるような、重要なアプリケーションに用いる。

2. ソフトリアルタイムクラス

このスレッドクラスは、例えその時間要件が満たされなかった場合においても、その品質が低下するだけのアプリケーションに用いる。

3. ベストエフォートクラス

このスレッドクラスは、その要件の中に一切の時間要件を含まないアプリケーションに用いる。

【0039】

スレッドは本リアルタイム処理システム内において処理を実行する実体である。スレッドには、そのスレッドが実行するプログラムが関連付けられている。各スレッドは、スレッドコンテキストと呼ぶ、それぞれのスレッドに固有の情報を保持している。スレッドコンテキストには、たとえば、プロセッサのレジスタの値や、スタックなどの情報が含まれている。

本リアルタイム処理システムにおいては、MPUSレッドとVPUSレッドの2種類のスレッドが存在する。これら2つのスレッドは、そのスレッドが実行されるプロセッサ (MPU11かVPU12) によって分類されており、スレッドとしてのモデルは全く同じである。VPUSレッドのスレッドコンテキストには、VPU12のローカルストレージ32の内容や、メモリコントローラ33の持つDMAコントローラの状態なども含む。

【0040】

複数のスレッドをグループとしてまとめたものを、スレッドグループと呼ぶ。

スレッドグループは、グループに含まれるスレッドすべてに対して同じ属性を与える、などの処理を効率よく簡単にできるメリットがある。ハードリアルタイムクラスまたはソフトリアルタイムクラスのスレッドグループは、密結合スレッドグループ (tightly coupled thread group) と疎結合スレッドグループ (loosely coupled thread group) の2種類に大別される。密結合スレッドグループ (tightly coupled thread group) と疎結合スレッドグループ (loosely coupled thread group) はスレッドグループに付加された属性情報 (結合属性情報) によって識別される。アプリケーションプログラム内のコードまたは上述の構成記述によってスレッドグループの結合属性を明示的に指定することができる。

密結合スレッドグループは互いに協調して動作する複数のスレッドの集合から構成されるスレッドグループである。すなわち、密結合スレッドグループは、そのグループに属するスレッド群が、お互いに密接に連携して動作することを示す。密接な連携とは、例えば、頻繁にスレッド間で通信あるいは同期処理などの相互作用 (interaction) を行ったり、あるいは、レイテンシ (latency) (遅延) の小さい相互作用を必要とする場合などである。一方、疎結合スレッドグループは、密結合スレッドグループに比べてそのグループに属するスレッド群間の密接な連携が不要であるスレッドグループであり、スレッド群はメモリ 14 上のバッファを介してデータ受け渡しのための通信を行う。

【0041】

(密結合スレッドグループ)

図25に示すように、密結合スレッドグループに属するスレッド群にはそれぞれ別のVPUが割り当てられ、各スレッドが同時に実行される。密結合スレッドグループに属するスレッドを、密結合スレッド (tightly coupled thread) と呼ぶ。この場合、密結合スレッドグループに属する密結合スレッドそれぞれの実行期間がそれら密結合スレッドの個数と同数のVPUそれぞれに対して予約され、それら密結合スレッドが同時に実行される。図25においては、ある密結合スレッドグループにスレッドA、Bの2つが密結合スレッドとして含まれており、それらスレッドA、BがそれぞれVPU0、VPU1によって同時に実行されている様子を示している。スレッドA、Bをそれぞれ別のVPUによって同時に実行

することを保証することにより、各スレッドは相手のスレッドが実行されている VPU のローカルストレージや制御レジスタを通じて相手のスレッドとの通信を直接的に行うことが出来る。図 26 は、スレッド A, B がそれぞれ実行される VPU0, VPU1 のローカルストレージを介してスレッド A, B 間の通信が実行される様子を示している。この場合、スレッド A が実行される VPU0 においては、そのスレッド A の EA 空間の一部に、通信相手のスレッド B が実行される VPU1 のローカルストレージ 32 に対応する RA 空間がマッピングされる。このマッピングのためのアドレス変換は、VPU0 のメモリコントローラ 33 内に設けられたアドレス変換ユニット 331 がセグメントテーブルおよびページテーブルを用いて実行する。スレッド B が実行される VPU1 においては、そのスレッド B の EA 空間の一部に、通信相手のスレッド A が実行される VPU0 のローカルストレージ 32 に対応する RA 空間がマッピングされる。このマッピングのためのアドレス変換は、VPU1 のメモリコントローラ 33 内に設けられたアドレス変換ユニット 331 がセグメントテーブルおよびページテーブルを用いて実行する。図 27 には、VPU0 上で実行されるスレッド A が自身の EA 空間にスレッド B が実行される VPU1 のローカルストレージ (LS1) 32 をマッピングし、VPU1 上で実行されるスレッド B が自身の EA 空間にスレッド A が実行される VPU0 のローカルストレージ (LS0) 32 をマッピングした様子が示されている。例えば、スレッド A はスレッド B に引き渡すべきデータがローカルストレージ LS0 上に準備できた時点で、そのことを示すフラグをローカルストレージ LS0 またはスレッド B が実行される VPU1 のローカルストレージ LS1 にセットする。スレッド B はそのフラグのセットに応答して、ローカルストレージ LS0 上のデータをリードする。

【0042】

このように、結合属性情報によって密結合関係にあるスレッドを特定できるようにすると共に、結合関係にあるスレッド A, B がそれぞれ別の VPU によって同時に実行されることを保証することにより、スレッド A, B 間の通信、同期に関するインタラクションをより軽量で且つ遅延無く行うことが可能となる。

【0043】

(疎結合スレッドグループ)

疎結合スレッドグループに属するスレッド群それぞれの実行時間は、それらスレッド群間の入出力関係によって決定され、たとえ実行順序の制約がないスレッド同士であってもそれらが同時に実行されることは保証されない。疎結合スレッドグループ属するスレッドを、疎結合スレッド (loosely coupled thread) と呼ぶ。図 2 8 においては、ある疎結合スレッドグループにスレッド C, D の 2 つが疎結合スレッドとして含まれており、それらスレッド C, D がそれぞれ V P U 0, V P U 1 によって実行されている様子を示している。図 2 8 に示すように、各スレッドの実行時間はばらばらになる。スレッド C, D 間の通信は、図 2 9 に示すように、メインメモリ 1 4 上に用意したバッファを介して行われる。スレッド C はローカルストレージ L S 0 に用意したデータを DMA 転送によってメインメモリ 1 4 上に用意したバッファに書き込み、スレッド D はその開始時に DMA 転送によってメインメモリ 1 4 上のバッファからローカルストレージ L S 1 にデータを読み込む。

【 0 0 4 4 】

(プロセスとスレッド)

プロセスは、図 3 0 に示すように、一つのアドレス空間と一つ以上のスレッドから構成される。一つのプロセスに含まれるスレッドの数と種類は、どのような組み合わせでも構わない。例えば、V P U スレッドのみから構成されるプロセスも構築可能であるし、V P U スレッドと M P U スレッドが混在するプロセスも構築可能である。スレッドがスレッド固有の情報としてスレッドコンテキストを保持しているのと同様に、プロセスもプロセス固有の情報としてプロセスコンテキストを保持する。このプロセスコンテキストには、プロセスに固有であるアドレス空間と、プロセスが含んでいる全スレッドのスレッドコンテキストが含まれる。プロセスのアドレス空間は、プロセスに属するすべてのスレッド間で共有することができる。一つのプロセスは、複数のスレッドグループを含むことができる。しかし、一つのスレッドグループが複数のプロセスに属することはできない。このため、あるプロセスに属するスレッドグループは、そのプロセスに固有であるということになる。本実施形態のリアルタイム処理システムにおいて、スレッ

ドを新しく生成する方式には、Thread first modelとAddress space first modelの2種類がある。Address space first modelは既存のOSで採用されていると同様の方式で、MPUSレッドにもVPUレッドにも適用できる。一方、Thread first modelはVPUレッドにしか適用できない方式で、本発明のリアルタイム処理システムに特有の方式である。Thread first modelでは、既存のスレッド（新しくスレッドを作りたいと思っている側のスレッド。新しく作るスレッドの親になるスレッドのこと。）は、まず新規スレッドが実行するプログラムを指定して、新規スレッドにプログラムの実行を開始させる。この時、プログラムはVPU12のローカルストレージに格納され、所定の実行開始番地から処理が開始される。この時点では、この新規スレッドにはアドレス空間が関連付けられていないので、自身のローカルストレージはアクセスできるが、メモリ14はアクセスできない。その後、新規スレッドは、必要に応じて自身でVPU実行環境のサービスを呼び出してアドレス空間を生成して関連付けたり、MPU11側の処理によってアドレス空間を関連付けられたりして、メモリ14にアクセスできるようになる。Address space first modelでは、既存のスレッドは、新しくアドレス空間を生成するか、あるいは既存のアドレス空間を指定して、そのアドレス空間に新規スレッドが実行するプログラムを配置する。そして新規スレッドにそのプログラムの実行を開始させる。Thread first modelのメリットは、ローカルストレージだけで動作するので、スレッドの生成やディスパッチや終了処理などのオーバーヘッドを小さくできることである。

【0045】

（スレッド群のスケジューリング）

次に、図31のフローチャートを参照して、VPU実行環境401によって実行されるスケジューリング処理について説明する。VPU実行環境401内のスケジューラは、スケジュール対象のスレッド群にスレッドグループ単位で付加されている結合属性情報に基づいて、スレッド間の結合属性をチェックし（ステップS121）、各スレッドグループ毎にそのスレッドグループが密結合スレッドグループおよび疎結合スレッドグループのいずれであるかを判別する（ステップS122）。結合属性のチェックは、プログラムコード中のスレッドに関する記

述あるいは上述の構成記述 117 中のスレッドパラメータを参照することによって行われる。このようにして、密結合スレッドグループおよび疎結合スレッドグループをそれぞれ特定することにより、スケジュール対象のスレッド群は密結合スレッドグループと疎結合スレッドグループとに分離される。

【0046】

密結合スレッドグループに属するスレッド群に対するスケジューリングは次のように行われる。すなわち、VPU 実行環境 401 内のスケジューラは、スケジュール対象のスレッド群から選択された密結合スレッドグループに属するスレッド群がそれぞれ別の VPU によって同時に実行されるように、その密結合スレッドグループに属するスレッド群と同数の VPU それぞれの実行期間を予約し、スレッド群をそれら予約した VPU それぞれに同時にディスパッチする（ステップ S123）。そして、スケジューラは、各スレッドが実行される VPU 内のアドレス変換ユニット 331 を用いて、各スレッドの EA 空間の一部に、協調して相互作用を行う相手となる他のスレッドが実行される VPU のローカルストレージに対応する RA 空間をマッピングする（ステップ S124）。一方、スケジュール対象のスレッド群から選択された疎結合スレッドグループに属する疎結合スレッド群については、スケジューラは、それらスレッド群間の入出力関係に基づいてそれらスレッド群を 1 以上の VPU に順次ディスパッチする（ステップ S125）。

【0047】

（ローカルストレージのマッピング）

本実施形態のリアルタイム処理システムにおいて、MPU スレッドと VPU スレッドの間、あるいは VPU スレッドと他の VPU スレッドの間で、何らかの通信や同期を行いながら協調して動作を行う場合には、協調相手の VPU スレッドのローカルストレージにアクセスする必要がある。たとえば、より軽量で高速な同期機構は、ローカルストレージ上に同期変数を割り付けて実装する。そのため、ある VPU 12 のローカルストレージを、他の VPU 12 あるいは MPU 11 のスレッドが直接アクセスする必要がある。図 4 に示す例のように、各 VPU 12 のローカルストレージが実アドレス空間に割り付けられている場合、セグメン

トテーブルやページテーブルを適切に設定すれば、相手のVPU12のローカルストレージを直接アクセスすることができる。しかしこの場合に、大きく2つの問題が生じる。

【0048】

第1の問題は、VPUスレッドのディスパッチ先VPU12の変更に関する問題である。図32のように、VPUスレッドAとBが存在し、それぞれVPU0とVPU1で動いているとする。そして、このスレッドAとBはお互いのスレッドと協調したいので、お互いのスレッドのLS（ローカルストレージ）を、自分のEA空間にマッピングしているとする。また、VPU0, 1, 2のLS0, 1, 2はそれぞれ図32のようにRA空間に存在するとする。この時、VPUスレッドAが、自分のEA空間にマッピングしているのは、VPUスレッドBが動いているVPUのLS、つまり、VPU1のLSであるLS1である。逆に、VPUスレッドBが、自分のEA空間にマッピングしているのは、VPUスレッドAが動いているVPUのLS、つまり、VPU0のLSであるLS0である。その後、VPU実行環境の中のスケジューラによって、VPUスレッドAを実行するVPUがディスパッチされて、VPUスレッドAは、VPU2で動くことになったとする。この時、もはやVPUスレッドAはVPU0では動いていないので、VPUスレッドBが、自分のEA空間にマッピングしているVPU0のLSは、意味がなくなる。この場合、スレッドBが、スレッドAのディスパッチ先VPUが変更になったことを知らなくてもいいように、システムは何らかの方法でLS0にマッピングされていたEA空間のアドレスをLS2にマッピングして、スレッドBから、スレッドAのローカルストレージとしてVPU2のLSであるLS2が見えるようにする必要がある。

【0049】

第2の問題は、物理VPUと論理VPUの対応関係の問題である。VPUをVPUスレッドに割り当てるまでには、実際には、2つのレベルがある。一つは論理VPUのVPUスレッドへの割り当てであり、もう一つが物理VPUの論理VPUへの割り当てである。物理VPUとは、仮想計算機OS301が管理している物理的なVPU12である。そして、論理VPUとは、仮想計算機OS301

がゲストOS割り当てた、論理的なVP Uのことである。この関係は図14にも示している。たとえば、VP U実行環境401が論理的なVP Uを管理する場合、図32の例で、VP Uスレッドの割り当て対象となるVP Uは論理VP Uである。

【0050】

図33は、この2つのレベルの割り当ての概念を示している。直前に説明した第1の問題は、図33の上の段に位置する、VP Uスレッドの論理VP Uへの割り当て問題に相当する。第2の問題である物理VP Uの論理VP Uへの割り当て問題は、下の段に位置する割り当てに相当する。図33では、4つの物理VP Uから、3つのVP Uを選択し、3つの論理VP Uに割り当てていることを示している。もし、この物理VP Uと論理VP Uの対応関係が変わった場合、VP Uスレッドの論理VP Uへの割り当てが変更になっていなくても、適切な設定の変更が必要となる。例えば、変更後の論理VP UのLSに対するアクセスが、正しい物理VP UのLSを指すように、LSに対応するページテーブルエントリを入れ換える、などである。

【0051】

ある時刻に、図34のように、物理VP U1, 2, 3が論理VP U0, 1, 2にそれぞれ割り当てられているとする。そして、論理VP U1はVP UスレッドAに、そして論理VP U2はVP UスレッドBに割り当てられていたとする。そして、VP UスレッドAとBは、それぞれ、お互いに、相手の動作している物理VP UのLSを自分のEA空間にマッピングしているとする。VP UスレッドAのEA空間にはVP UスレッドBが実行されている物理VP U3のLS3が、そしてVP UスレッドBのEA空間にはVP UスレッドAが実行されている物理VP U2のLS2がマッピングされている。その後、ある時刻に、仮想計算機OS301によって、論理VP U0, 1が物理VP U0, 1に、再割り当てされたとする。すると、今までVP UスレッドAが動作していた論理VP U1は、物理VP U2から物理VP U1へと変化する。論理VP UのVP Uスレッドへの割り当ては変化していないが、物理VP Uと論理VP Uの対応関係が変化したことになる。このため、VP UスレッドBがEA空間にマッピングしている、VP Uスレ

ッドAの動作しているVP UのLSを、LS 2からLS 1に変更し、正しくアクセスできるようにする必要がある。

【0052】

これらの2つの問題を解決するために、本実施形態のリアルタイム処理システムでは、スレッドから見たEA空間の固定アドレスに、必ず相手のスレッドを実行しているVP Uのローカルストレージがマップされて見えるように仮想記憶機構を制御する。すなわち、VP Uスケジューラによる論理VP Uのディスパッチ時、および仮想計算機OS等による物理VP Uと論理VP Uの対応関係の切り替え時に、適宜ページテーブルやセグメントテーブルを書き換えることで、VP U上で動作しているスレッドからは、いつも同じ番地に相手のスレッドを実行しているVP Uのローカルストレージが見えるようにする。

【0053】

まず、2つのスレッドのEA空間の関係について説明する、2つのスレッドのEA空間は、次の3つのいずれかのパターンで共有あるいは非共有になっている。

1. 共有EA型： 2つのスレッド1, 2がセグメントテーブルもページテーブルも共有している（図35）

2. 共有VA型： 2つのスレッド1, 2は、ページテーブルは共有するが、セグメントテーブルは共有せず、それぞれが持っている（図36）

3. 非共有型： 2つのスレッド1, 2はページテーブルもセグメントテーブルも共有せず、それぞれが持っている（図37）

以下、1の共有EA型を例に、VP Uのローカルストレージをどのようにマップするように制御するかについて説明する。

まず、図38に示すように、VA空間上に各論理VP Uに対応した領域を設け、そこに、その論理VP Uが対応付けられている物理VP Uのローカルストレージがマップされるように、ページテーブルを設定する。この例の場合、物理VP U0, 1, 2がそれぞれ論理VP U0, 1, 2に対応付けられている状態を示している。次に、スレッドAからはスレッドBを実行しているVP Uのローカルストレージが、固定アドレスであるセグメントaの領域に見えるように、セグメン

トテーブルを設定する。また、スレッドBからはスレッドAを実行している論理VPUのローカルストレージが、固定アドレスであるセグメントbに見えるように、セグメントテーブルを設定する。この例では、スレッドAは論理VPU2で、スレッドBは論理VPU1で実行している状況を示している。ここで、VPU実行環境401のスケジューラが、スレッドBを論理VPU0にディスパッチしたとする。この時、VPU実行環境401は、図39に示すように、スレッドAからは固定アドレスであるセグメントaを通して、スレッドBを現在実行している論理VPU0のローカルストレージに見えるように、VPU実行環境401はセグメントテーブルを自動的に書き換える。

さらにここで、たとえば仮想計算機OS3.01がゲストOSのディスパッチをしたため、物理VPUと論理VPUの対応が変化したとする。このとき、たとえば図40に示すように、VPU実行環境401は、ページテーブルを書き換えて、VA空間上に固定されている論理VPUのローカルストレージの領域が、正しい物理VPUのローカルストレージの領域を指すようにする。図40の例では、物理VPU1, 2, 3が論理VPU0, 1, 2に対応するように変更されたため、ページテーブルを書き換えて、現在の正しいマッピングになるようにしている。

【0054】

このように、VPU実行環境401のスケジューラのディスパッチによって、スレッドを実行する論理VPUが変更になった場合には、EA空間からVA空間へのマッピングを行っているセグメントテーブルを書き換えて、第1の問題を解決している。また、仮想計算機OS3.01などによって、物理VPUと論理VPUの対応が変更になった場合は、VA空間からRA空間へのマッピングを行っているページテーブルを書き換えて、第2の問題を解決している。

このようにして、相互作用を行う相手のスレッドが実行されるプロセッサに応じて、実効アドレス空間にマッピングされる、相手のスレッドに対応するプロセッサのローカルメモリが自動的に変更することにより、各スレッドは相手のスレッドがディスパッチされるプロセッサを意識することなく、相手のスレッドとの相互作用を効率よく行うことが出来る。よって、複数のスレッドを効率よく並列に

実行することが可能となる。

【0055】

以上、共有EA型の場合の例を説明したが、2の共有VA型、3の非共有型についても、セグメントテーブルまたはページテーブルを書き換えることにより、同様に第1の問題および第2の問題を解決することができる。

【0056】

上記の第1および第2の問題を解決する別の方法について述べる。ここでも、共有EA型の場合を例に説明する。図41に示すように、協調して動作する複数のVPUSレッドがある場合、それらのスレッドを実行するVPUのローカルストレージを、セグメント上に連続してマップするように、ページテーブルとセグメントテーブルを設定する。図41の例の場合、スレッドAは物理VPU2で、スレッドBは物理VPU0で実行されており、それぞれのVPUのローカルストレージが同一のセグメントに連続して配置されるように、ページテーブルとセグメントテーブルを設定している。ここで、VPU実行環境401のスケジューラによってスレッドを実行する論理VPUがディスパッチされたり、仮想計算機OS301等によって物理VPUと論理VPUの対応が変更になった場合には、それぞれの変更がスレッドAおよびスレッドBに対して隠蔽されるように、ページテーブルを書き換えて、VA空間とRA空間のマップを変更する。たとえば図42は、スレッドAを実行しているVPUが物理VPU1に、スレッドBを実行しているVPUが物理VPU3に変更になった場合のマッピングを示している。この変更が行われても、スレッドAおよびスレッドBからは、固定したアドレスを持つセグメント内の、所定の領域をアクセスすることで、常に相手のスレッドを実行しているVPUのローカルストレージをアクセスすることができる。

【0057】

次に、図43のフローチャートを参照して、VPU実行環境401によって実行されるアドレス管理処理の手順について説明する。VPU実行環境401は、各スレッドのEA空間上の固定アドレスに、相手スレッドを実行しているVPUのローカルストレージに対応するRA空間をマッピングする（ステップS201）。この後、VPU実行環境401は、相手スレッドのディスパッチ先VPUの

変更あるいは論理 V P U と物理 V P U の対応関係の変更に起因して、相手スレッドが実行される V P U が変更されたかどうかを判別する（ステップ S 2 0 2）。相手スレッドが実行される V P U が変更されたならば、V P U 実行環境 4 0 1 は、セグメントテーブルまたはページテーブルの内容を書き換えて、各スレッドの E A 空間上の固定アドレスにマッピングされているローカルストレージを、相手スレッドが実行される V P U に合わせて変更する（ステップ S 2 0 3）。

【0058】

これまでの例では、蜜結合スレッドグループのように、互いに V P U によって実行中のスレッド間で、相手のスレッドを実行している V P U のローカルストレージをアクセスする方式を説明した。しかし、疎結合スレッドグループなど、協調して動作するスレッドが必ずしも同時に V P U に割り当てられて実行していない場合も存在する。そのような場合でも、E A 空間上には相手のスレッドを実行している V P U 1 2 のローカルストレージをマップする領域は存在するので、その領域を以下のように用いて対処する。

【0059】

第 1 の方法： 相手のスレッドが実行中で無い場合には、そのスレッドに対応する V P U のローカルストレージをマップする領域にアクセスすると、スレッドは相手のスレッドが実行開始するまで待たされるようにする。

第 2 の方法： 相手のスレッドが実行中で無い場合には、そのスレッドに対応する V P U のローカルストレージをマップする領域にアクセスすると、スレッドは例外発生やエラーコードによって、その旨を知る。

【0060】

第 3 の方法： スレッドの終了時に、そのスレッドを最後に実行していたときのローカルストレージの内容をメモリに保存しておき、そのスレッドに対応付けられたローカルストレージを指すページテーブルあるいはセグメントテーブルのエントリからは、そのメモリ領域を指すようにマッピングを制御する。この方式により、相手のスレッドが実行中でなくても、相手のスレッドに対応付けられたローカルストレージがあたかもあるように、スレッドの実行を続けることができる。図 4 4 および図 4 5 に具体例を示す。

①： いま、スレッドA、BがそれぞれVPU0、1で実行されており、スレッドBのEA空間には相手のスレッドAが実行されているVPU0のローカルストレージLS0がマッピングされているとする。

②： スレッドAの終了時には、スレッドAまたはVPU実行環境401は、スレッドAが実行されているVPU0のローカルストレージLS0の内容をメモリ14に保存する（ステップS211）。

③： VPU実行環境401は、スレッドBのEA空間にマッピングされている相手先スレッドAのローカルストレージのアドレス空間を、VPU0のLS0から、LS0の内容が保存されたメモリ14上のメモリ領域に変更する（ステップS212）。これにより、スレッドBは、相手のスレッドAが実行中でなくなった後も、その動作を継続することができる。

④： スレッドAに再びVPUが割り当てられたとき、VPU実行環境401は、メモリ14上のメモリ領域をスレッドAが実行されるVPUのローカルストレージに戻す（ステップS213）。たとえばスレッドAに再びVPU0が割り当てられたときは、メモリ14上のメモリ領域の内容は、VPU0のローカルストレージLS0に戻される。

⑤： VPU実行環境401は、スレッドBのEA空間にマッピングされている相手先スレッドAのローカルストレージのアドレス空間を、スレッドAが実行されるVPUのローカルストレージに変更する（ステップS214）。たとえばスレッドAに再びVPU0が割り当てられたときは、スレッドBのEA空間にマッピングされている相手先スレッドAのローカルストレージのアドレス空間は、VPU0のローカルストレージLS0に戻される。

【0061】

なお、スレッドAにVPU2が割り当てられたときは、メモリ14上のメモリ領域の内容は、VPU2のローカルストレージLS2に復元される。そして、スレッドBのEA空間にマッピングされている相手先スレッドAのローカルストレージのアドレス空間は、VPU2のローカルストレージLS2に変更される。

【0062】

なお、図1の計算機システムに設けられたMPU11と複数のVPU12は。

それらを 1 チップ上に混載した並列プロセッサとして実現することもできる。この場合も、MPU 1 1 によって実行される VPU 実行環境、あるいは特定の一つの VPU などによって実行される VPU 実行環境が、複数の VPU 1 2 に対するスケジューリングおよびアドレス管理を行うことが出来る。

【 0 0 6 3 】

また VPU 実行環境として動作するプログラムまたはその VPU 実行環境を含むオペレーティングシステムなどのプログラムをコンピュータ読み取り可能な記憶媒体に記憶することにより、その記憶媒体を通じて当該プログラムを、ローカルプロセッサをそれぞれ有する複数のプロセッサを含むコンピュータに導入して実行するだけで、本実施形態と同様の効果を得ることが出来る。

【 0 0 6 4 】

また、本発明は上記実施形態そのままに限定されるものではなく、実施段階ではその要旨を逸脱しない範囲で構成要素を変形して具体化できる。また、上記実施形態に開示されている複数の構成要素の適宜な組み合わせにより、種々の発明を形成できる。例えば、実施形態に示される全構成要素から幾つかの構成要素を削除してもよい。さらに、異なる実施形態にわたる構成要素を適宜組み合わせてもよい。

【 0 0 6 5 】

【発明の効果】

以上説明したように、本発明によれば、複数のプロセッサを用いて複数のスレッドを効率よく並列に実行することが可能となる。

【図面の簡単な説明】

【図 1】 本発明の一実施形態に係るリアルタイム処理システムを構成する計算機システムの例を示すブロック図。

【図 2】 同実施形態のリアルタイム処理システムに設けられた MPU および VPU それぞれの構成を示すブロック図。

【図 3】 同実施形態のリアルタイム処理システムで用いられる仮想アドレス変換機構の例を示す図。

【図 4】 同実施形態のリアルタイム処理システムにおける実アドレス空間

にマッピングされるデータの例を示す図。

【図 5】 同実施形態のリアルタイム処理システムにおける実効アドレス空間、仮想アドレス空間、実アドレス空間を説明するための図。

【図 6】 デジタルテレビ放送の受信機の構成を示すブロック図。

【図 7】 同実施形態のリアルタイム処理システムによって実行されるプログラムモジュールの構成の例を示す図。

【図 8】 図 7 のプログラムモジュール内に含まれる構成記述の例を示す図。

【図 9】 図 7 のプログラムモジュールに対応するプログラム間のデータの流れを示す図。

【図 10】 図 7 のプログラムモジュールが 2 つの V P U によって並列に実行される様子を示す図。

【図 11】 図 7 のプログラムモジュールが 2 つの V P U によってパイプライン形式で実行される様子を示す図。

【図 12】 同実施形態のリアルタイム処理システムにおけるオペレーティングシステムの実装形態の例を示す図。

【図 13】 同実施形態のリアルタイム処理システムにおけるオペレーティングシステムの実装形態の他の例を示す図。

【図 14】 同実施形態のリアルタイム処理システムにおける仮想計算機 O S とゲスト O S との関係を示す図。

【図 15】 同実施形態のリアルタイム処理システムにおいて複数のゲスト O S に時分割で資源が割り当てられる様子を示す図。

【図 16】 同実施形態のリアルタイム処理システムにおいてある特定のゲスト O S によって特定の資源が専有される様子を示す図。

【図 17】 同実施形態のリアルタイム処理システムにおいてスケジューラとして用いられる V P U 実行環境を示す図。

【図 18】 同実施形態のリアルタイム処理システムで用いられる仮想計算機 O S に V P U 実行環境を実装した例を示す図。

【図 19】 同実施形態のリアルタイム処理システムで用いられる一つのゲ

スト OS として V P U 実行環境を実装する例を示す図。

【図 2 0】 同実施形態のリアルタイム処理システムで用いられる複数のゲスト OS それぞれに V P U 実行環境を実装する例を示す図。

【図 2 1】 同実施形態のリアルタイム処理システムで用いられる一つのゲスト OS に V P U 実行環境を実装する例を示す図。

【図 2 2】 同実施形態のリアルタイム処理システムで用いられる M P U 側 V P U 実行環境と V P U 側 V P U 実行環境を説明するための図。

【図 2 3】 同実施形態のリアルタイム処理システムで用いられる V P U 側 V P U 実行環境によって実行される処理手順を示すフローチャート。

【図 2 4】 同実施形態のリアルタイム処理システムで用いられる M P U 側 V P U 実行環境によって実行される処理手順を示すフローチャート。

【図 2 5】 同実施形態のリアルタイム処理システムにおいて密結合スレッドグループに属するスレッド群がそれぞれ別のプロセッサによって同時に実行される様子を示す図。

【図 2 6】 同実施形態のリアルタイム処理システムにおける密結合スレッド間の相互作用を説明するための図。

【図 2 7】 同実施形態のリアルタイム処理システムにおいて各密結合スレッドの実効アドレス空間に相手のスレッドが実行される V P U のローカルストレージがマッピングされる様子を示す図。

【図 2 8】 同実施形態のリアルタイム処理システムにおける疎結合スレッドグループに属するスレッド群に対するプロセッサの割り当てを説明するための図。

【図 2 9】 同実施形態のリアルタイム処理システムにおける疎結合スレッド間の相互作用を説明するための図。

【図 3 0】 同実施形態のリアルタイム処理システムにおけるプロセスとスレッドとの関係を説明するための図。

【図 3 1】 同実施形態のリアルタイム処理システムにおけるスケジューリング処理の手順を示すフローチャート。

【図 3 2】 同実施形態のリアルタイム処理システムにおけるローカルスト

レージのマッピングに関する第 1 の問題を説明するための図。

【図 3 3】 同実施形態のリアルタイム処理システムにおける物理 V P U と論理 V P U との関係を示す図。

【図 3 4】 同実施形態のリアルタイム処理システムにおけるローカルストレージのマッピングに関する第 2 の問題を説明するための図。

【図 3 5】 同実施形態のリアルタイム処理システムにおける実効アドレス空間共有モデルを示す図。

【図 3 6】 同実施形態のリアルタイム処理システムにおける仮想アドレス空間共有モデルを示す図。

【図 3 7】 同実施形態のリアルタイム処理システムにおける非共有モデルを示す図。

【図 3 8】 同実施形態のリアルタイム処理システムにおけるローカルストレージのマッピング変更を説明するための第 1 の図。

【図 3 9】 同実施形態のリアルタイム処理システムにおけるローカルストレージのマッピング変更を説明するための第 2 の図。

【図 4 0】 同実施形態のリアルタイム処理システムにおけるローカルストレージのマッピング変更を説明するための第 3 の図。

【図 4 1】 同実施形態のリアルタイム処理システムにおけるローカルストレージのマッピング変更を説明するための第 4 の図。

【図 4 2】 同実施形態のリアルタイム処理システムにおけるローカルストレージのマッピング変更を説明するための第 5 の図。

【図 4 3】 同実施形態のリアルタイム処理システムにおいてローカルストレージのマッピング変更を行うために実行されるアドレス管理処理の手順を示すフローチャート。

【図 4 4】 同実施形態のリアルタイム処理システムにおいて実行されるローカルストレージとメモリとの間のマッピング変更を説明するための図。

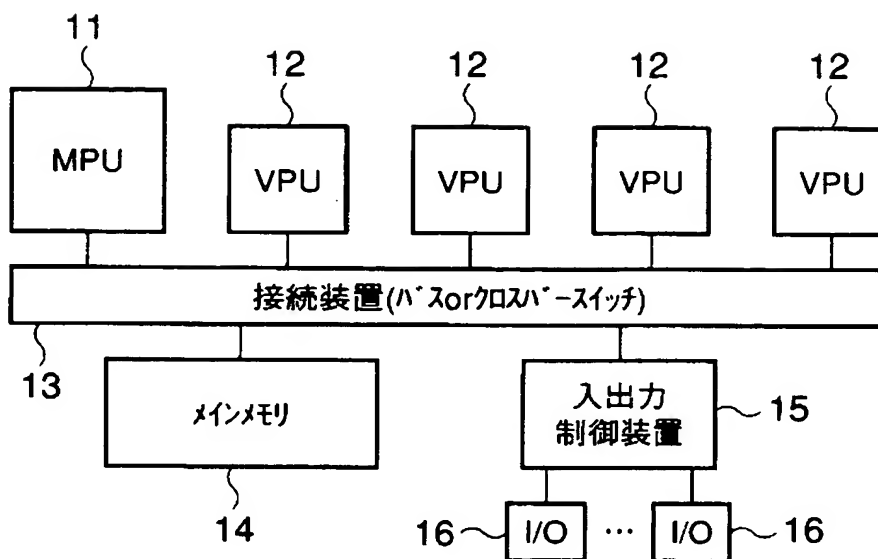
【図 4 5】 同実施形態のリアルタイム処理システムにおいて実行されるローカルストレージとメモリとの間のマッピング変更処理の手順を示すフローチャート。

【符号の説明】

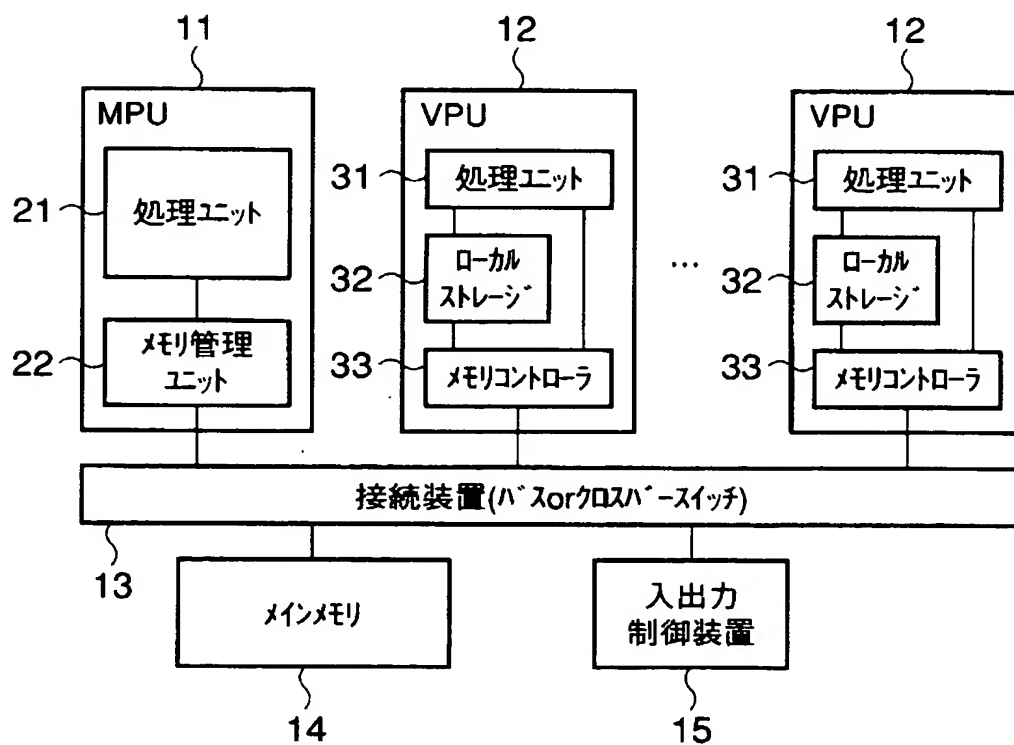
1 1…M P U (Master Processing Unit)、1 2…V P U (Slave Processing Unit)、1 4…メインメモリ、2 1…処理ユニット、2 2…メモリ管理ユニット、3 1…処理ユニット、3 2…ローカルストレージ、3 3…メモリコントローラ、5 0…セグメントテーブル、6 0…ページテーブル、1 0 0…プログラムモジュール、1 1 7…構成記述、3 3 1…アドレス変換ユニット、4 0 1…V P U 実行環境。

【書類名】 図面

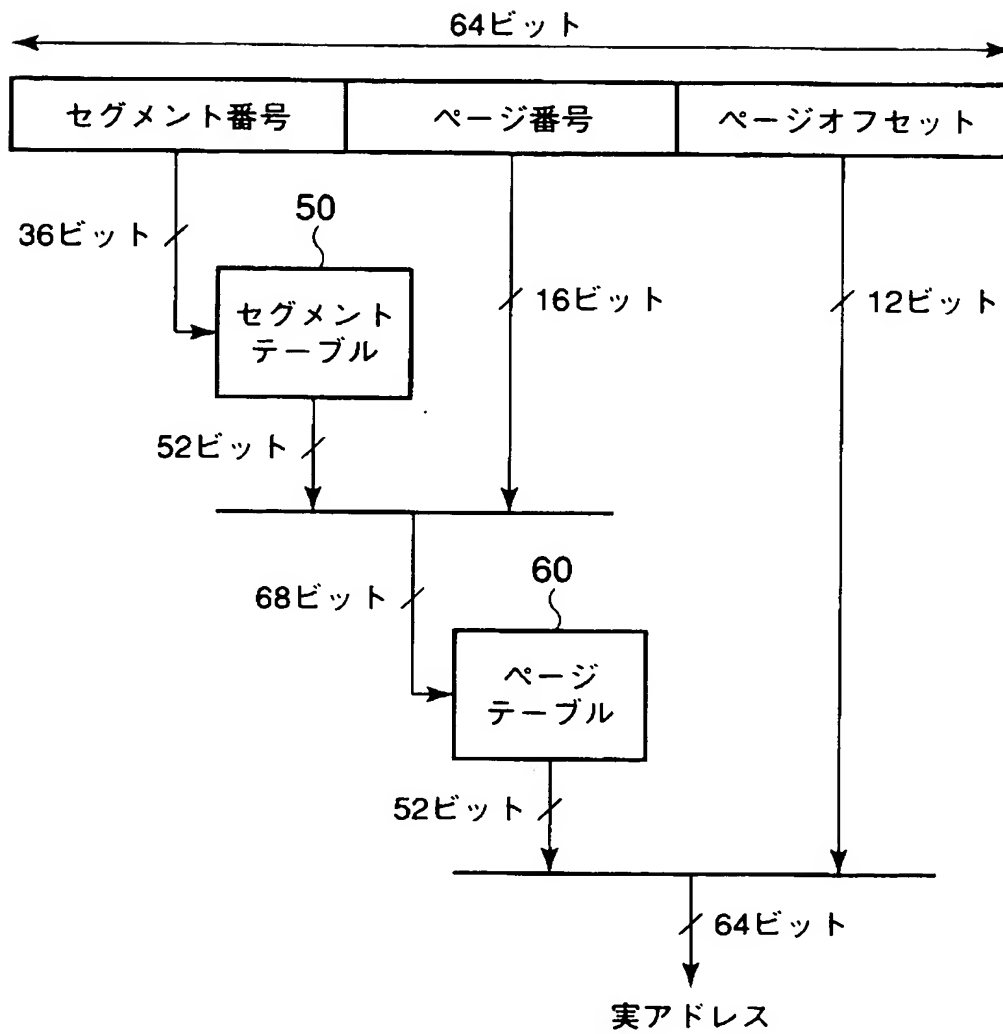
【図 1】



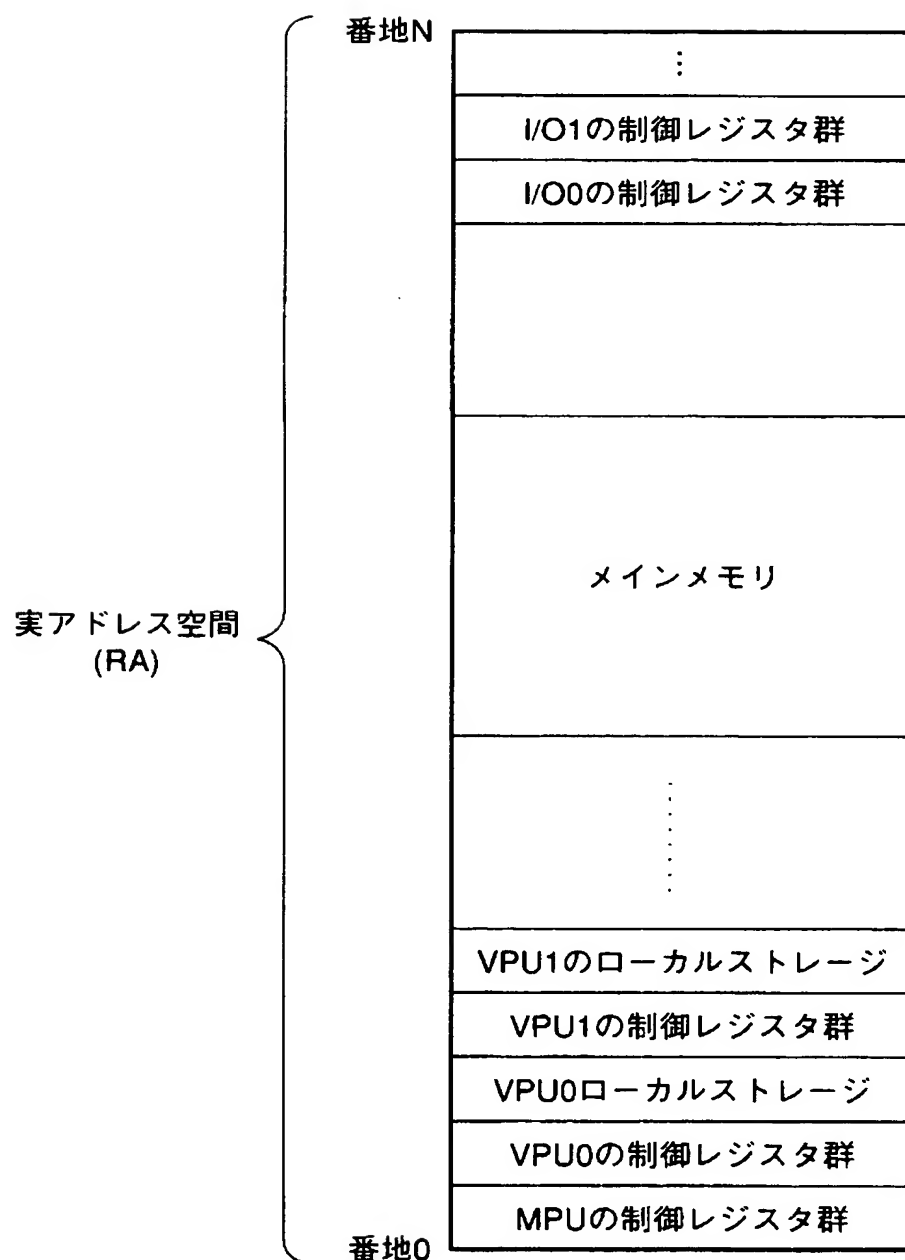
【図 2】



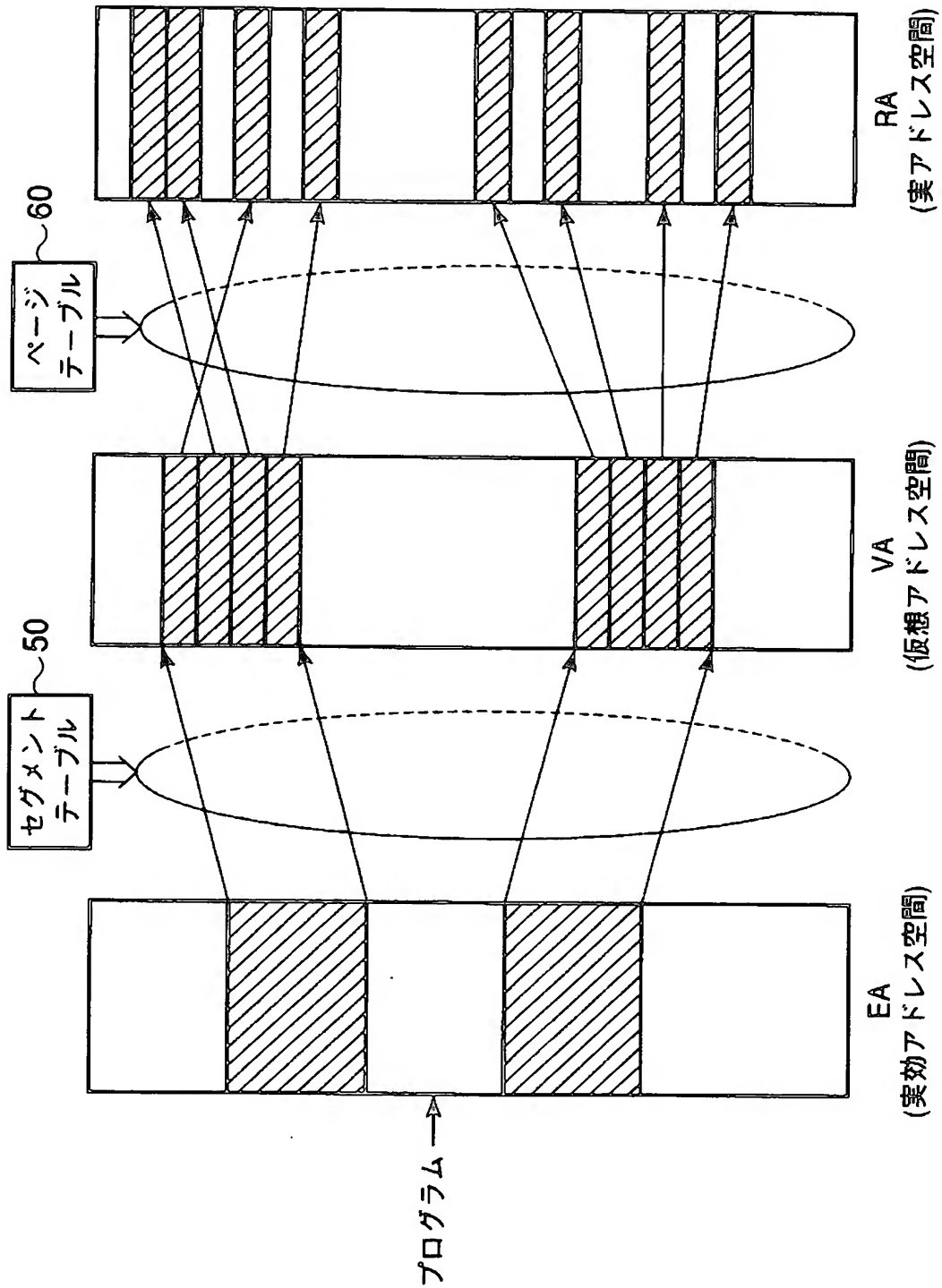
【図 3】



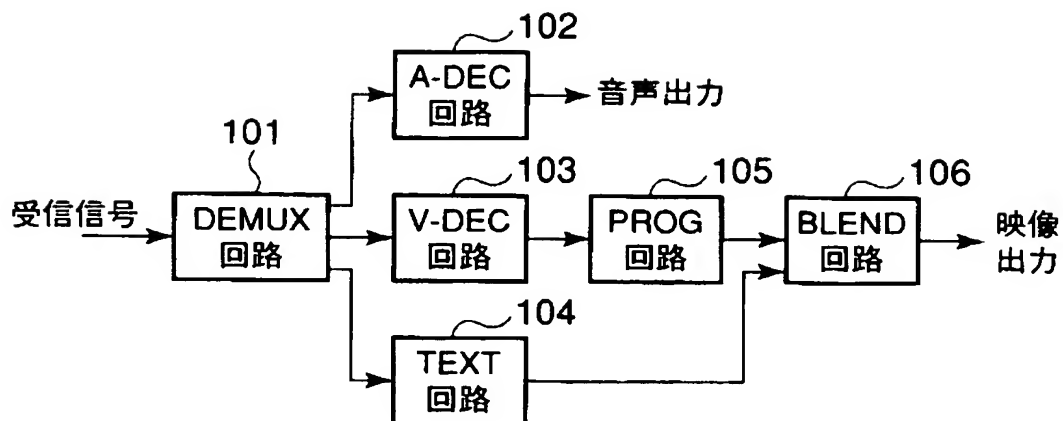
【図 4】



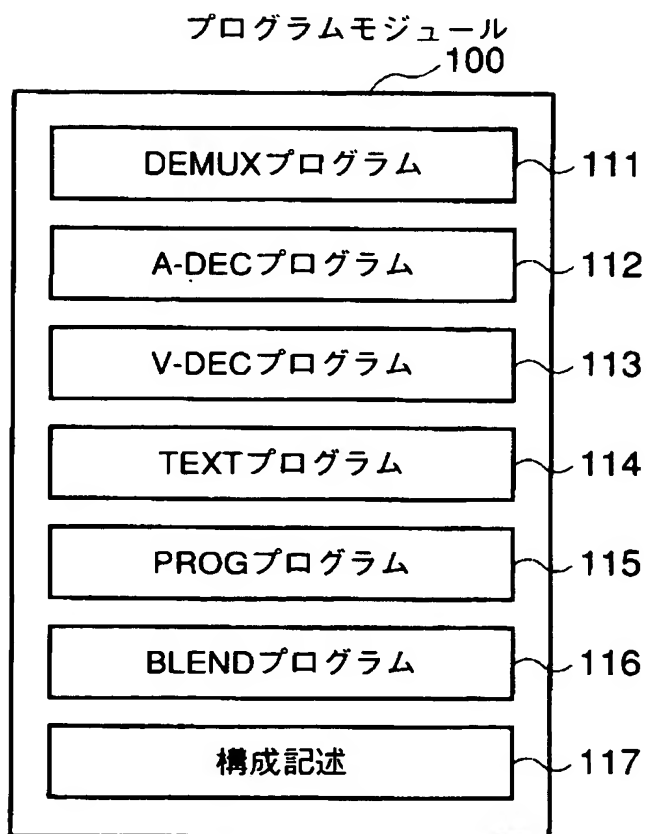
【図 5】



【図 6】



【図 7】

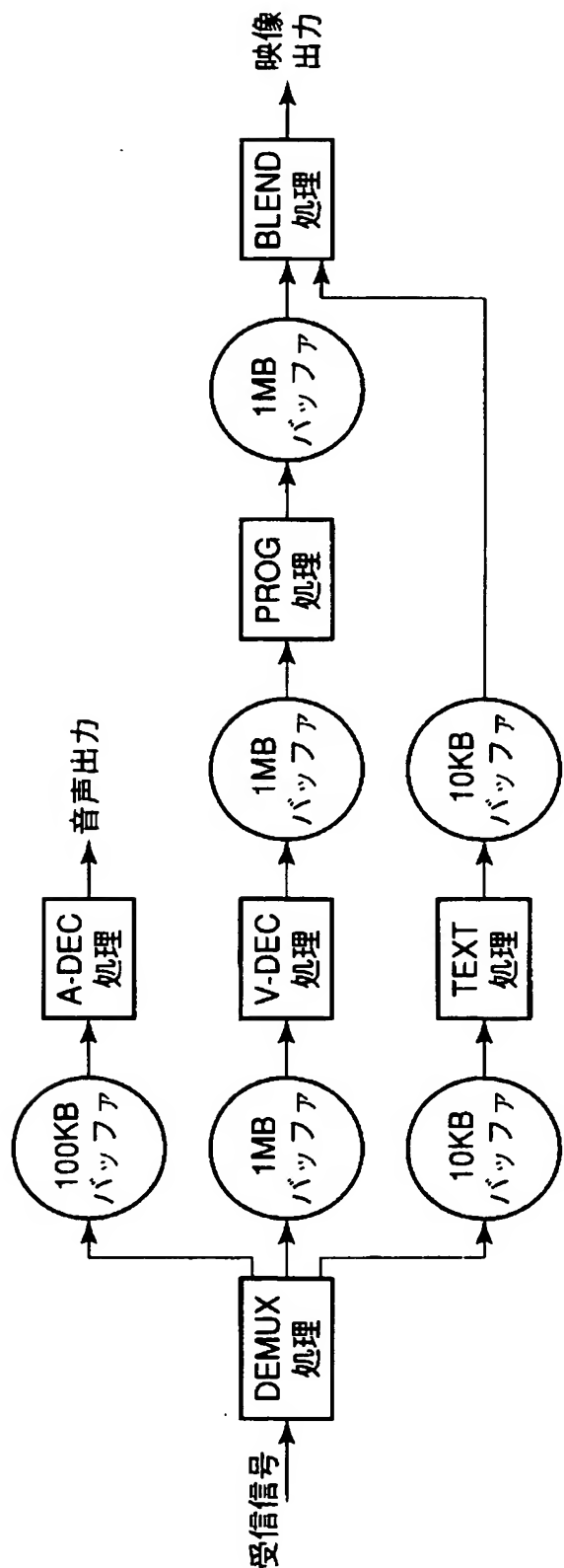


【図 8】

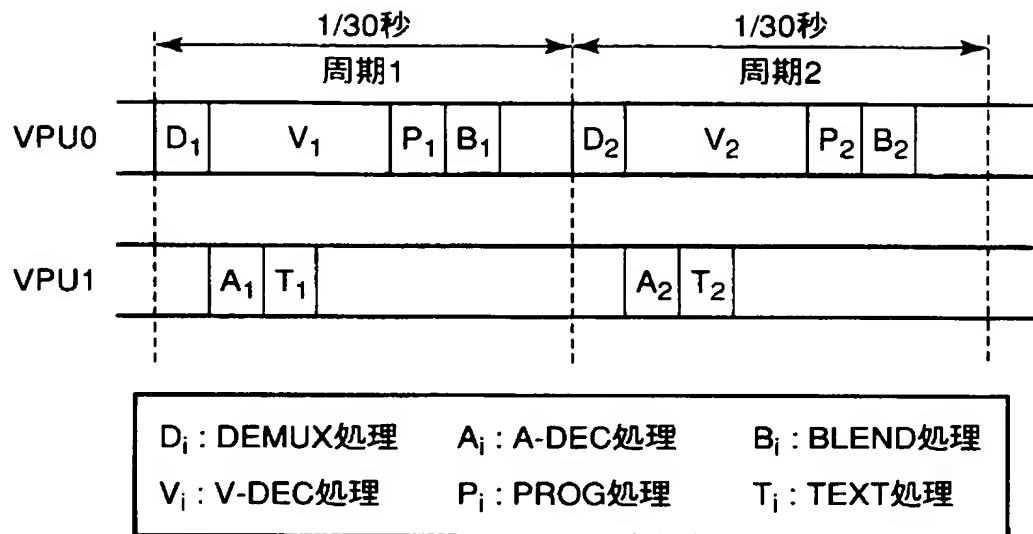
117 構成記述の例

番号	プログラム	入力	出力	コスト	バッファ
①	DEMUX	受信信号	② ③ ④	5	100KB 1MB 10KB
②	A-DEC	①	音声出力	10	——
③	V-DEC	①	⑤	50	1MB
④	TEXT	①	⑥	5	10KB
⑤	PROG	③	⑥	20	1MB
⑥	BLEND	④ ⑤	映像出力	10	——
スレッドパラメータ					
その他					

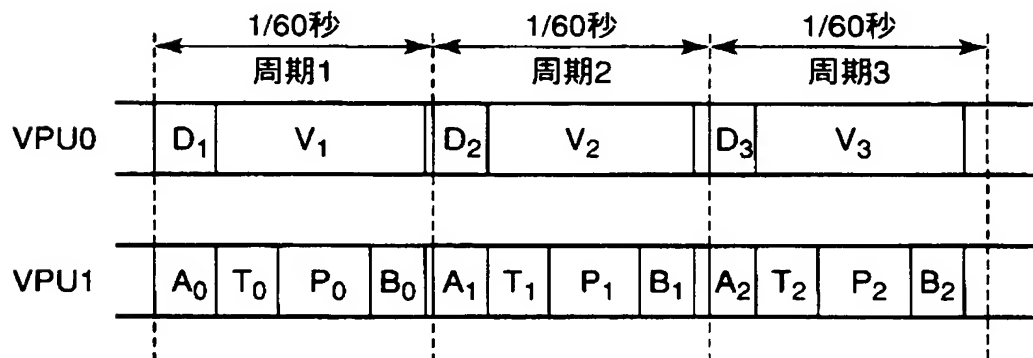
【図 9】



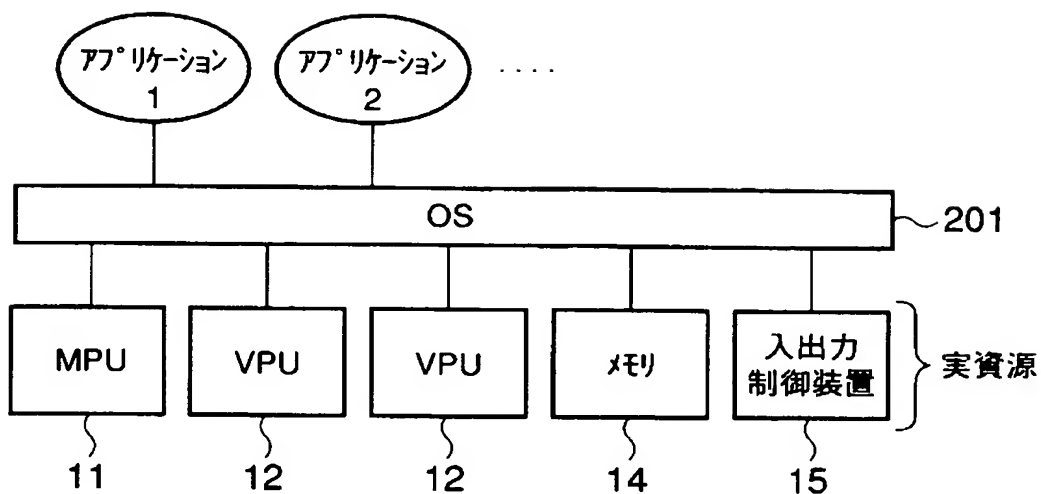
【図 10】



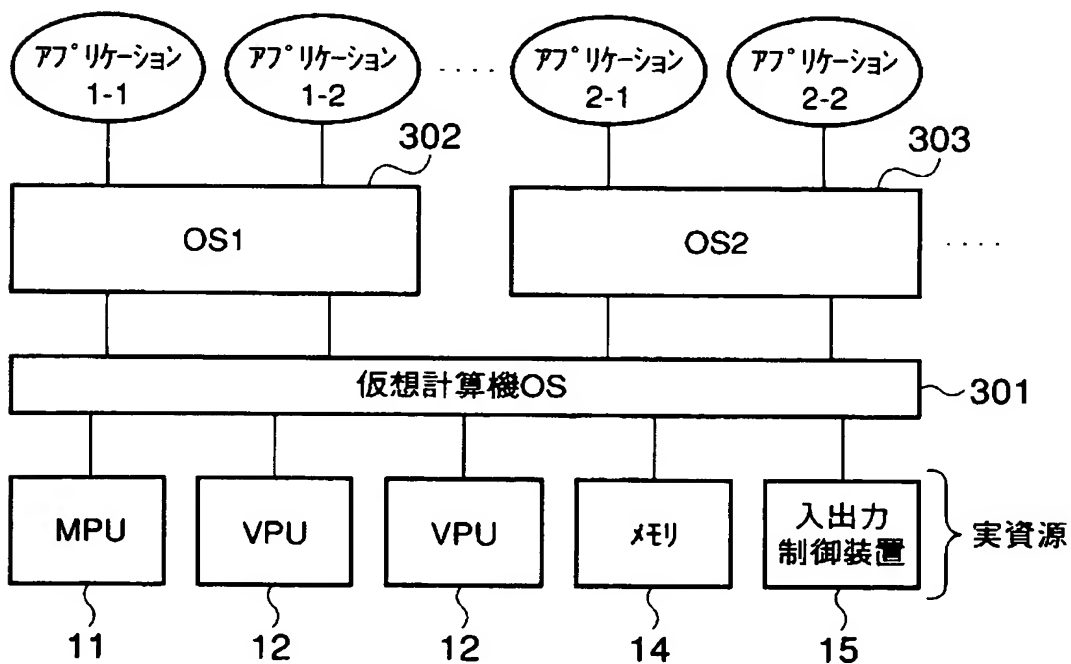
【図 11】



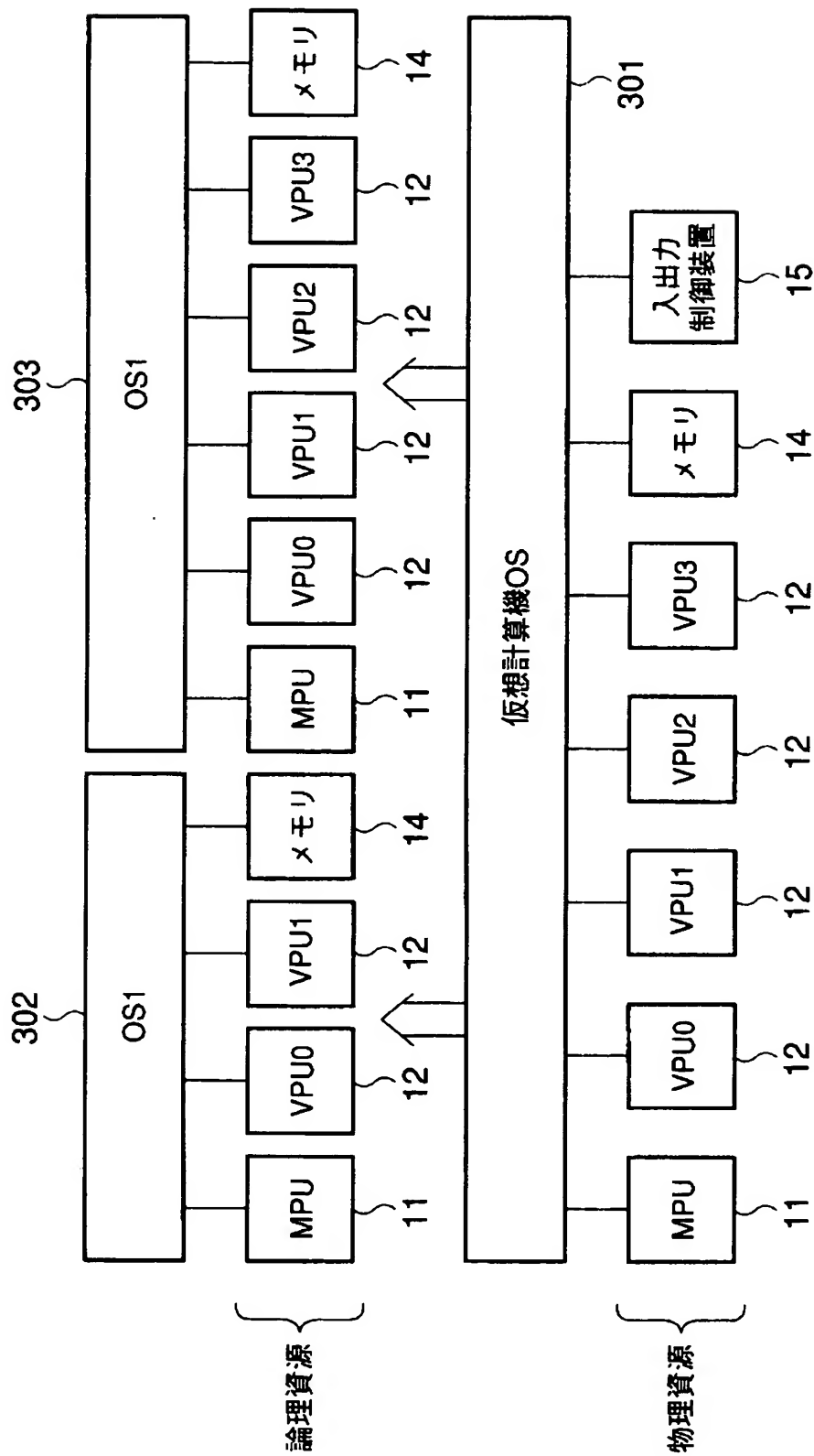
【図 12】



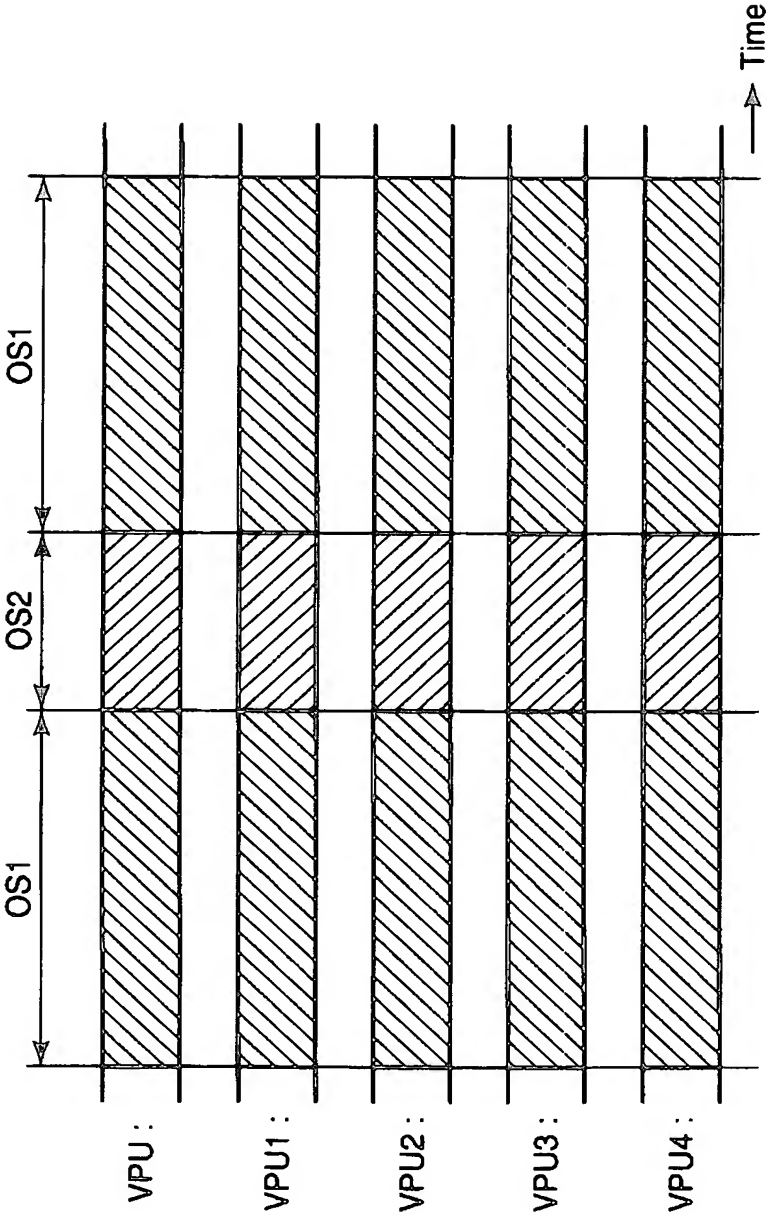
【図 13】



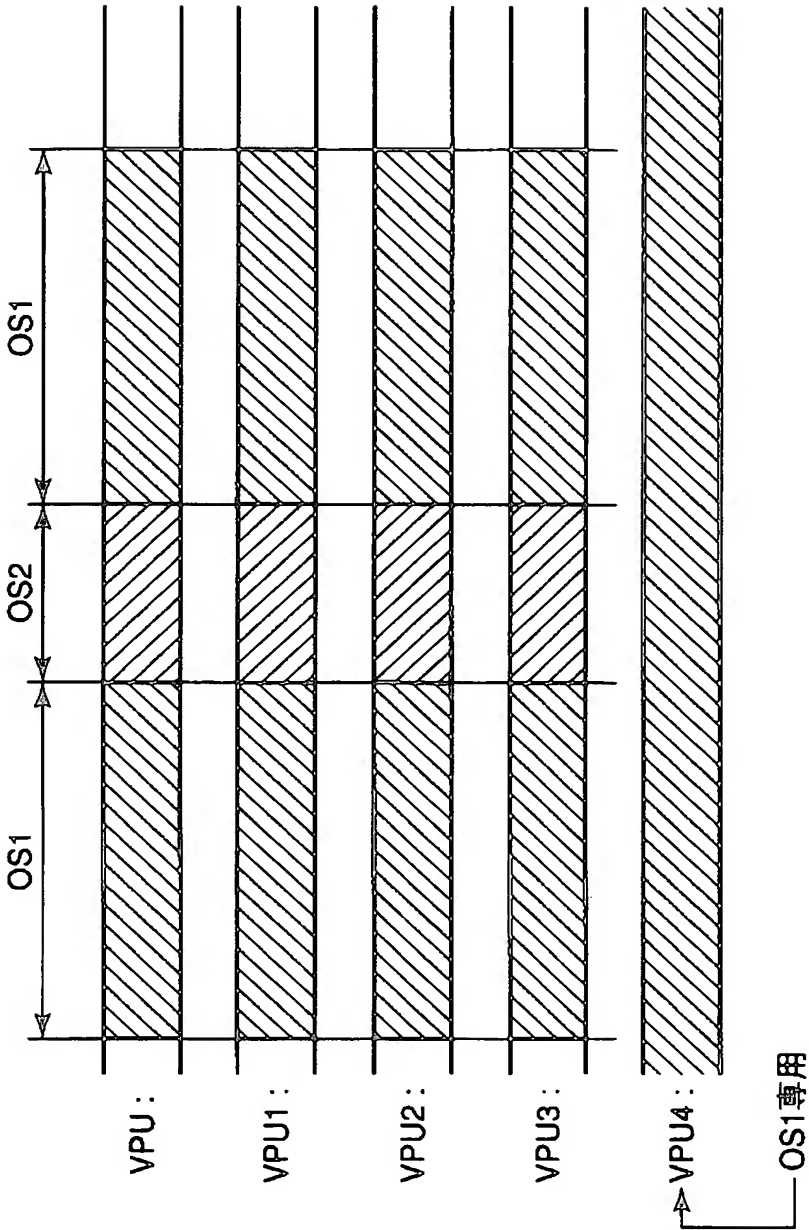
【図 14】



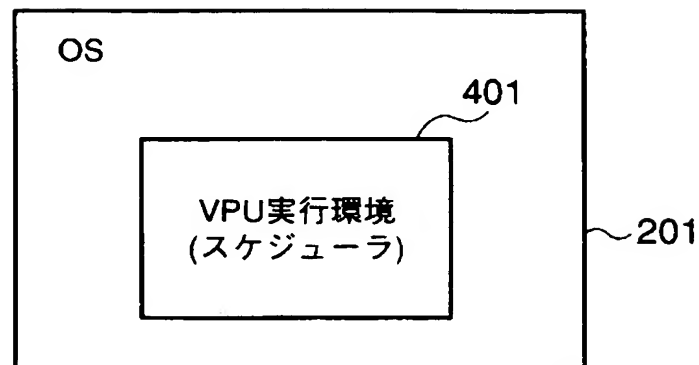
【図 1 5】



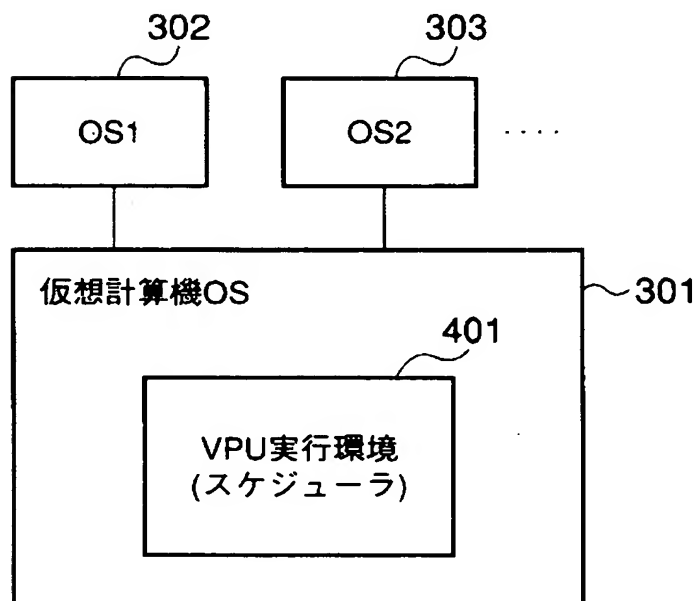
【図 1 6】



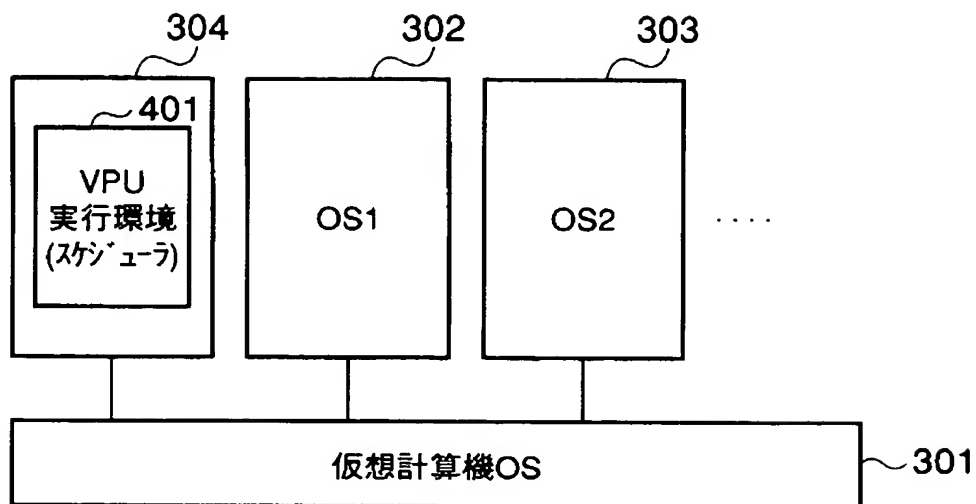
【図 17】



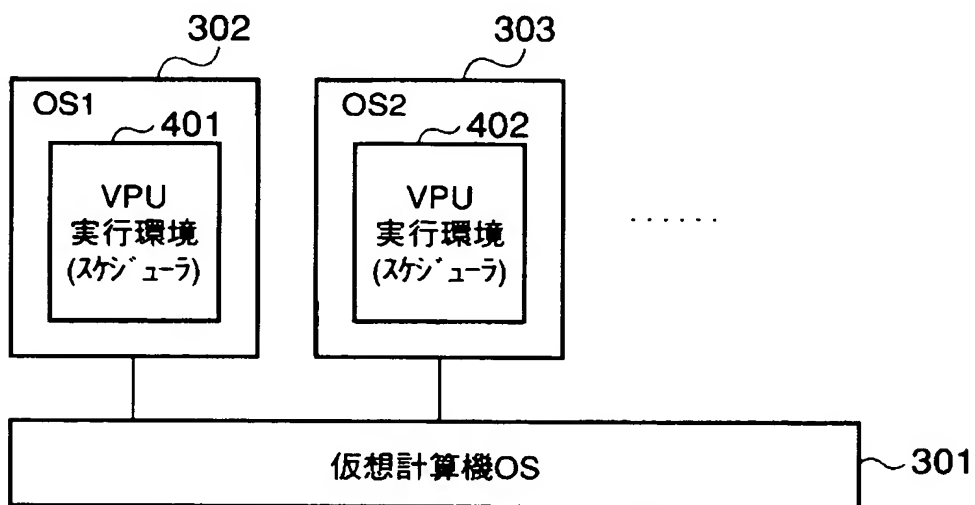
【図 18】



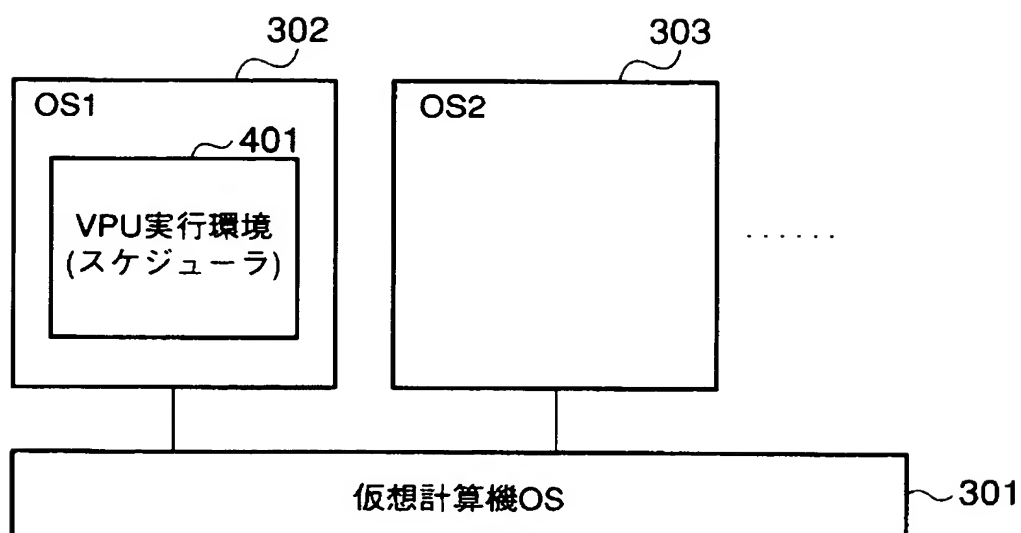
【図 19】



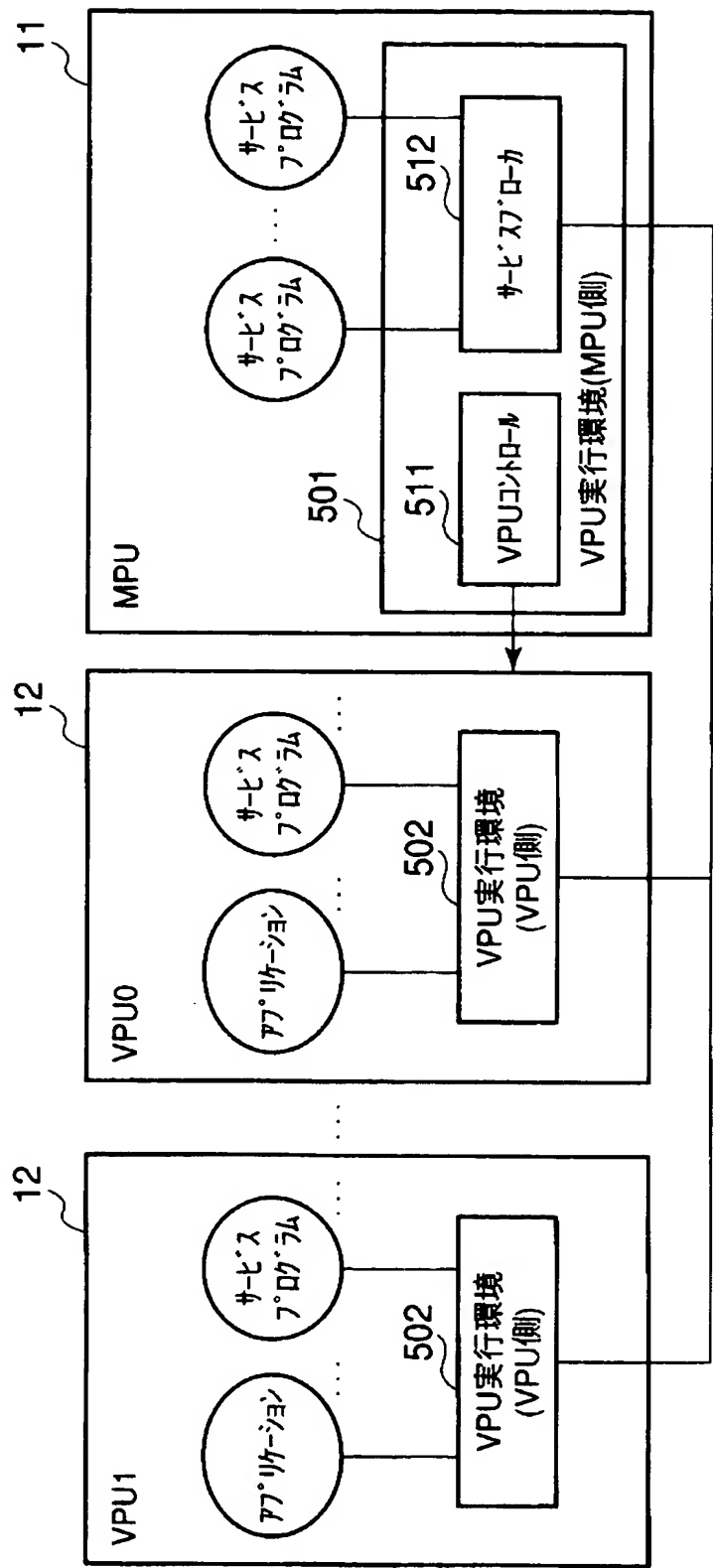
【図 20】



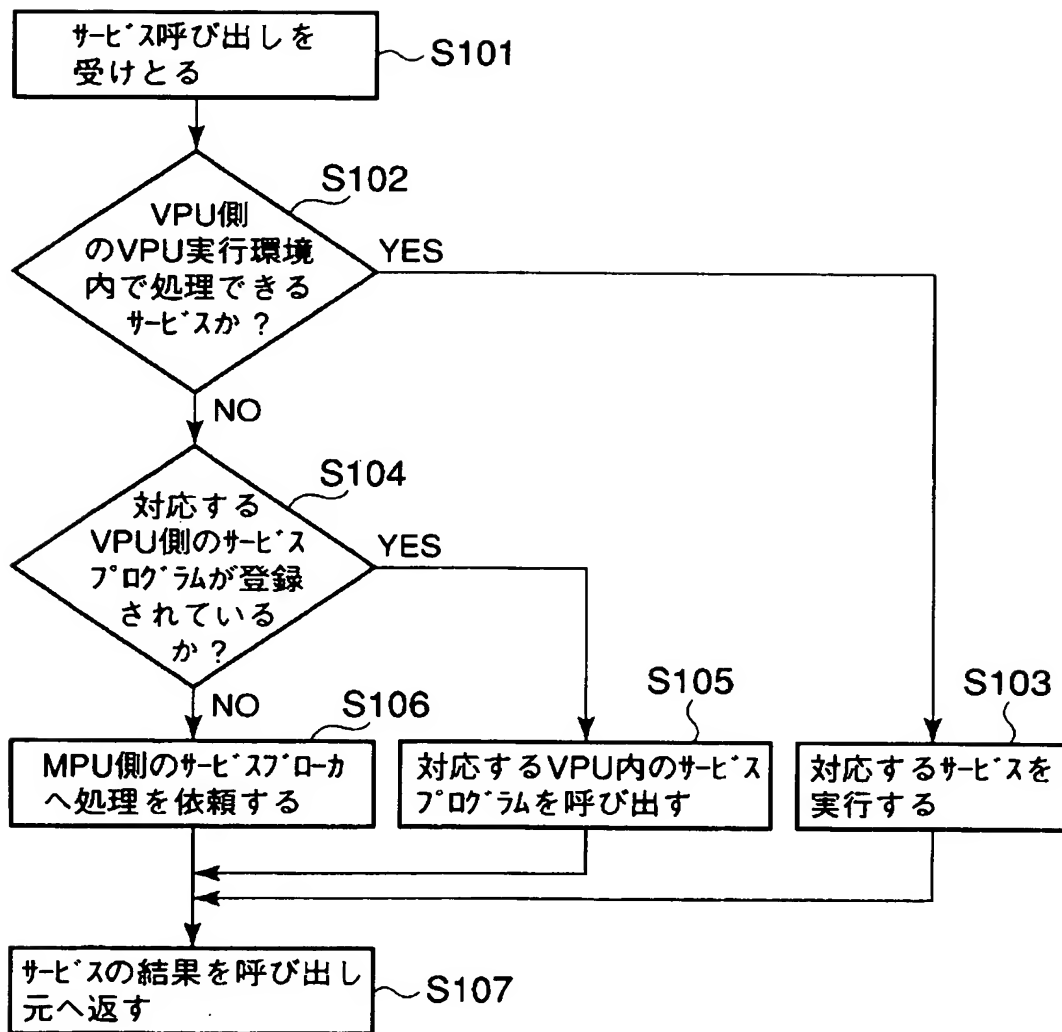
【図 21】



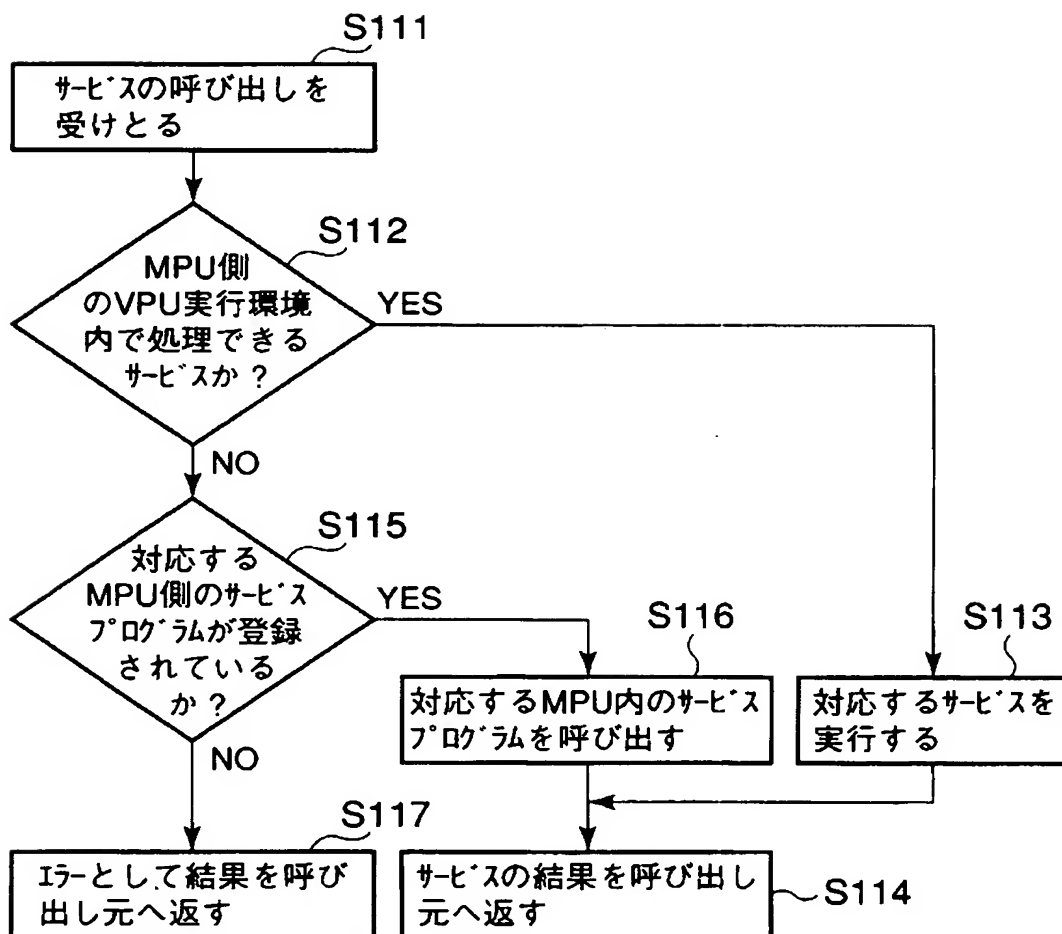
【図 22】



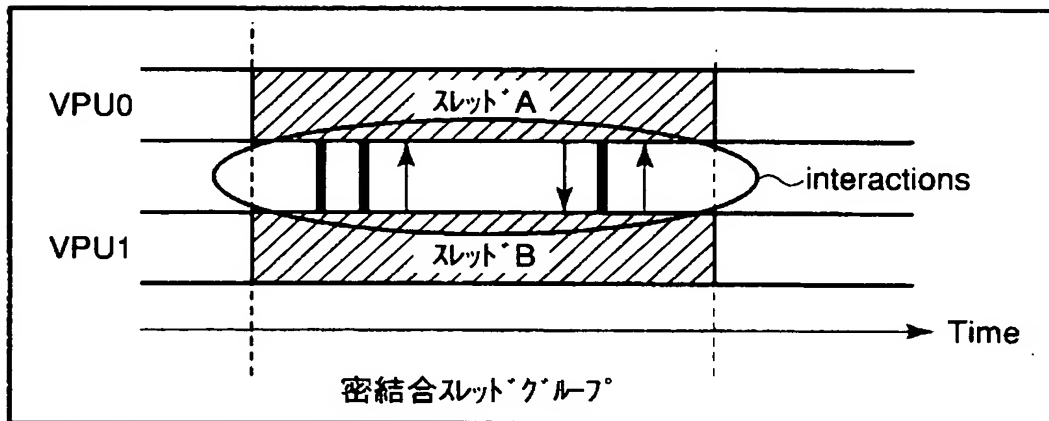
【図 23】



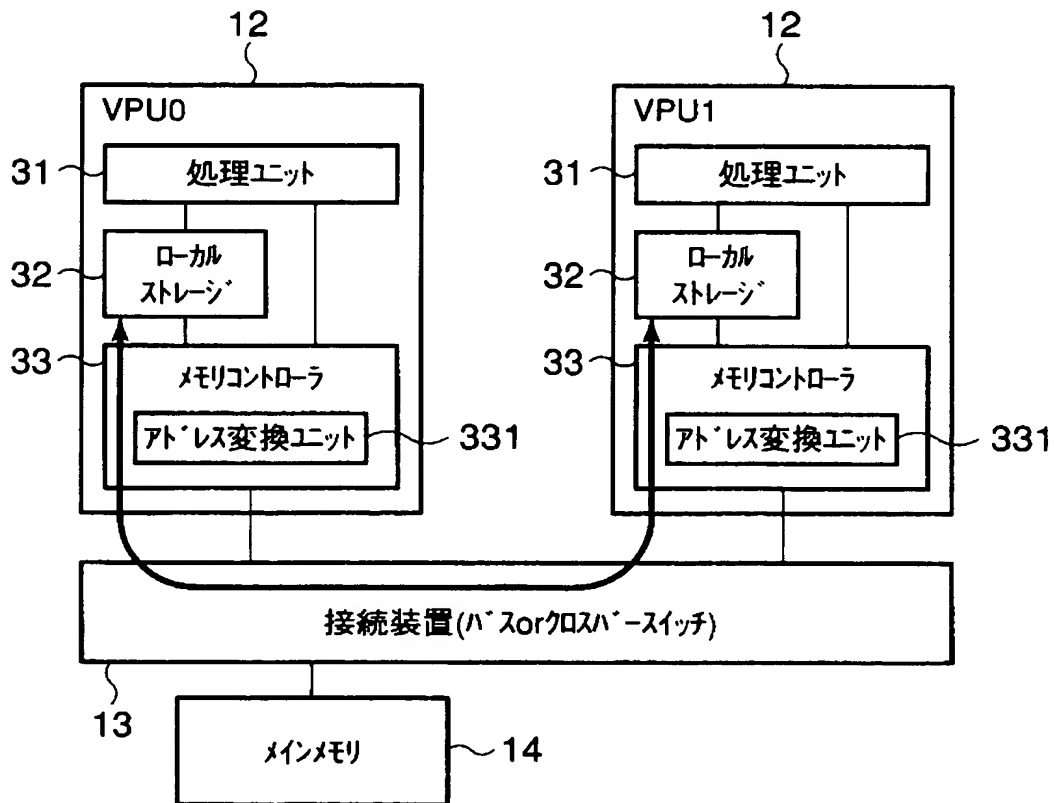
【図 24】



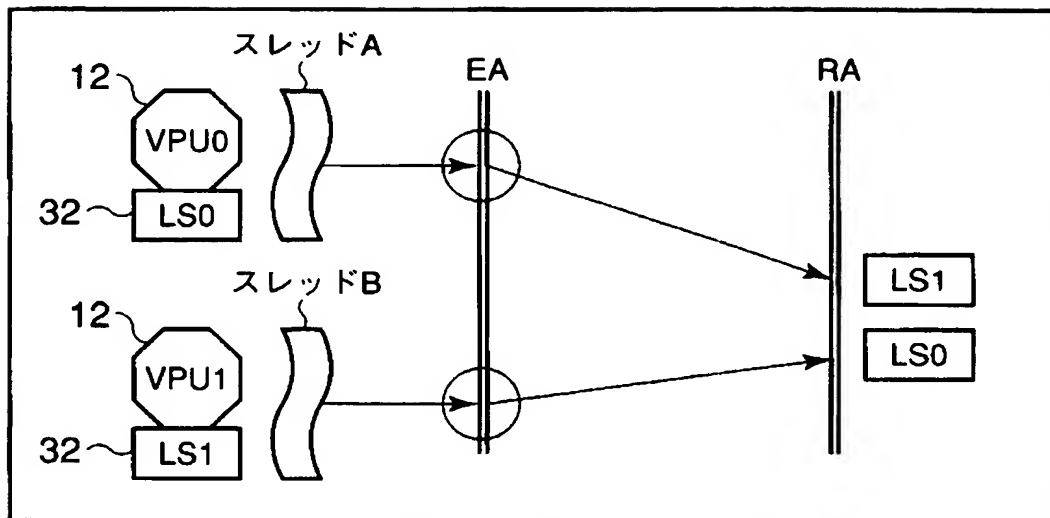
【図 25】



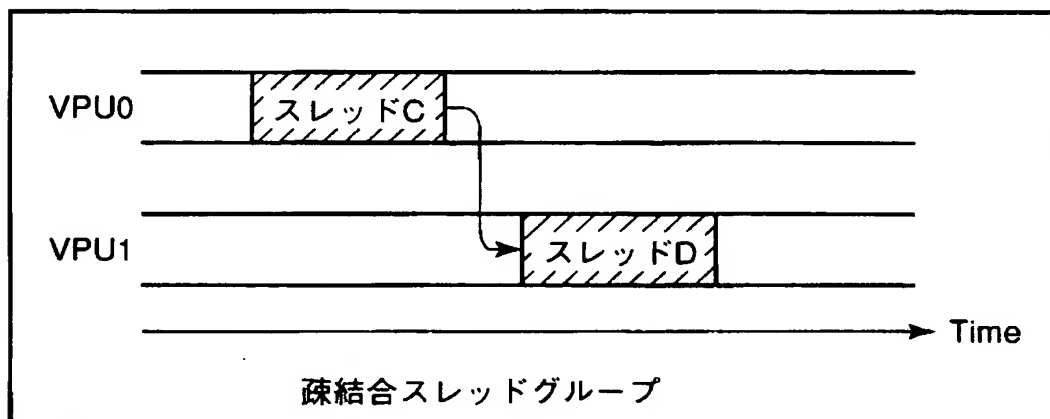
【図 26】



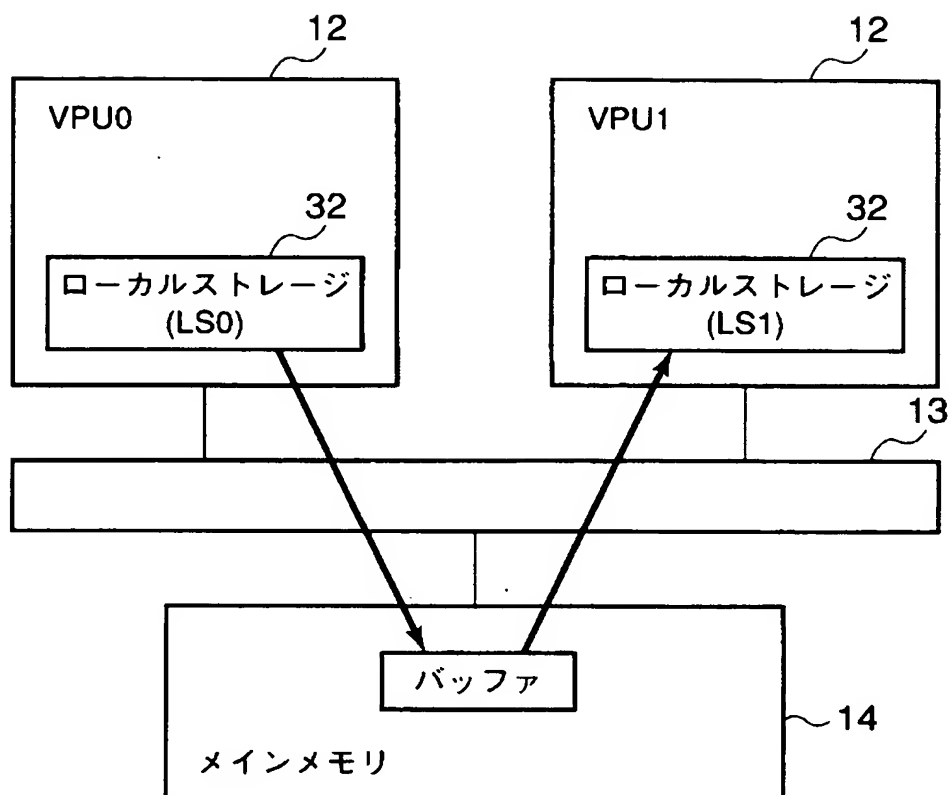
【図 27】



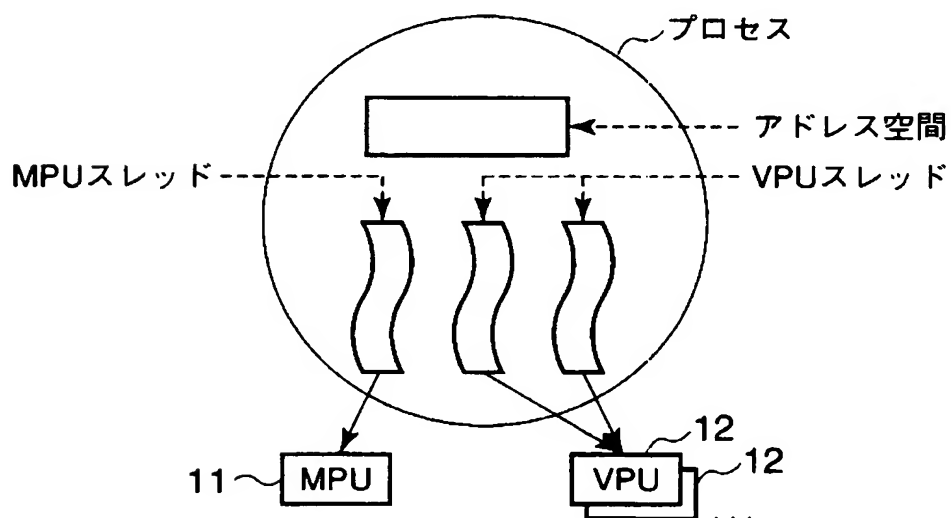
【図 28】



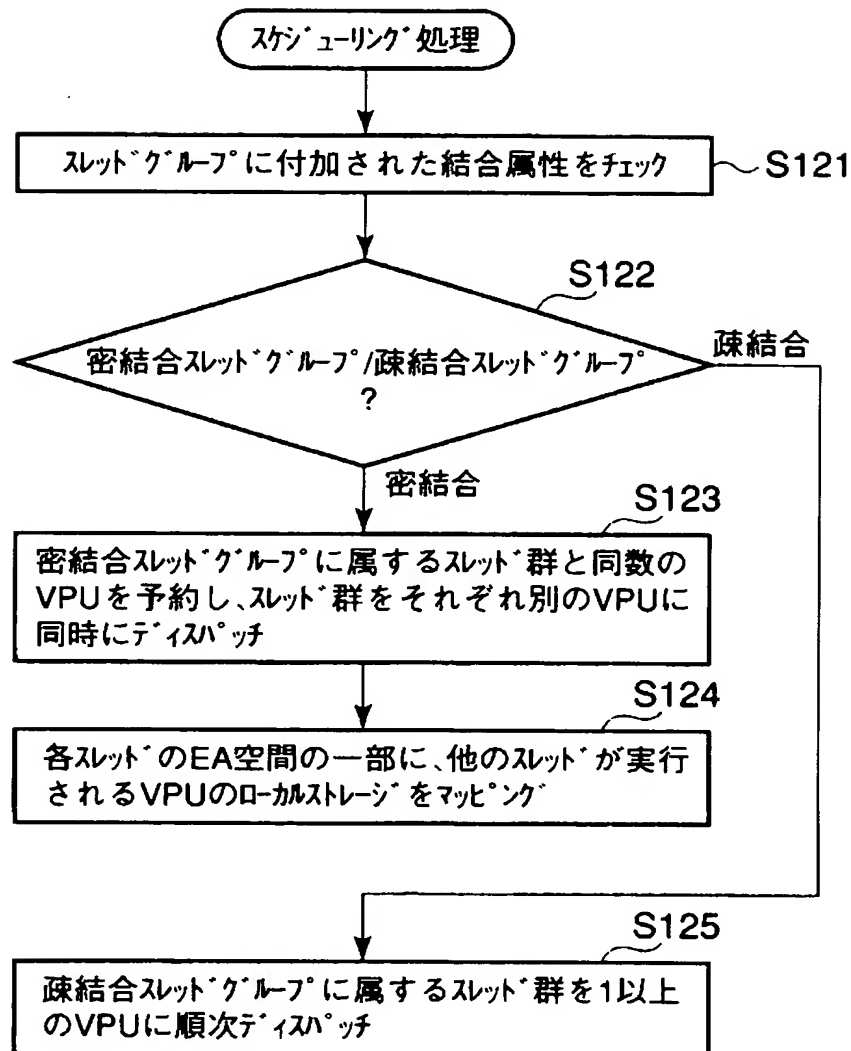
【図 29】



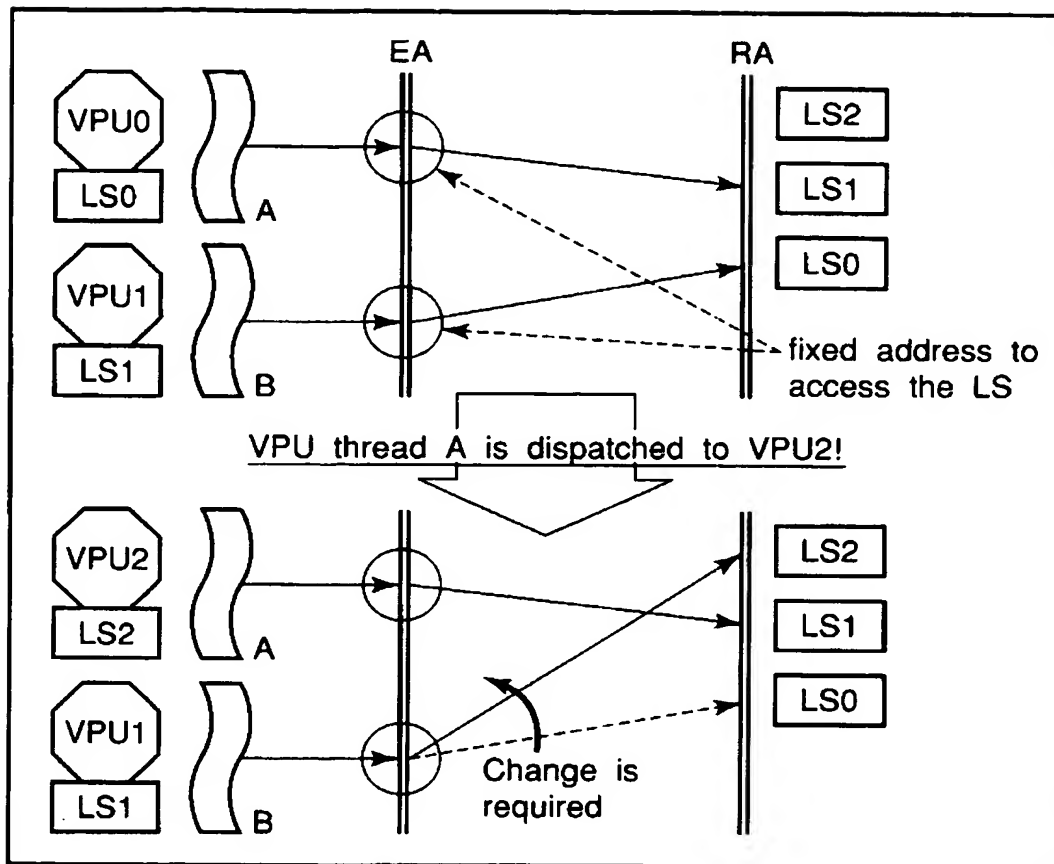
【図 30】



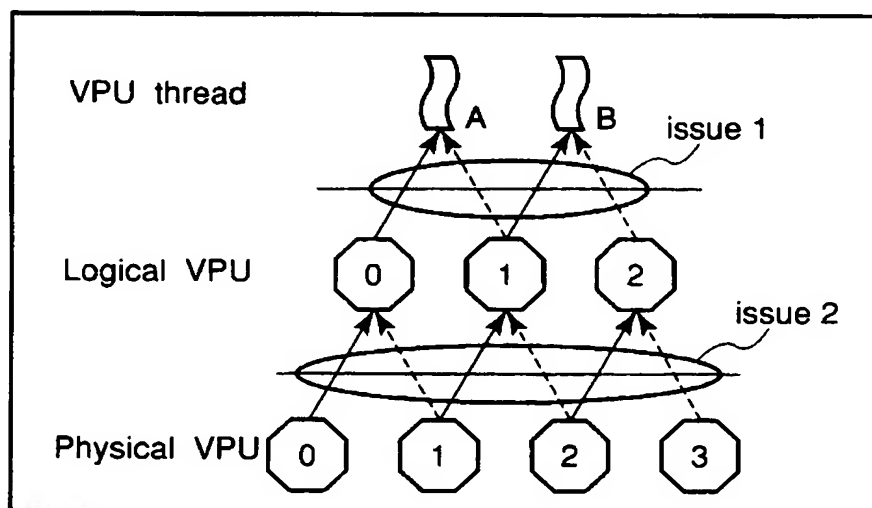
【図 31】



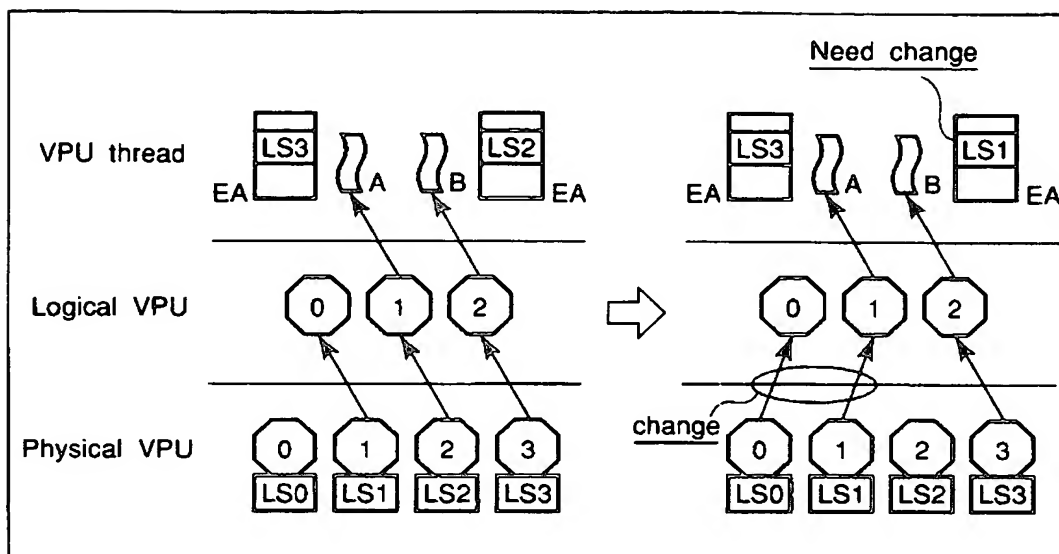
【図 3 2】



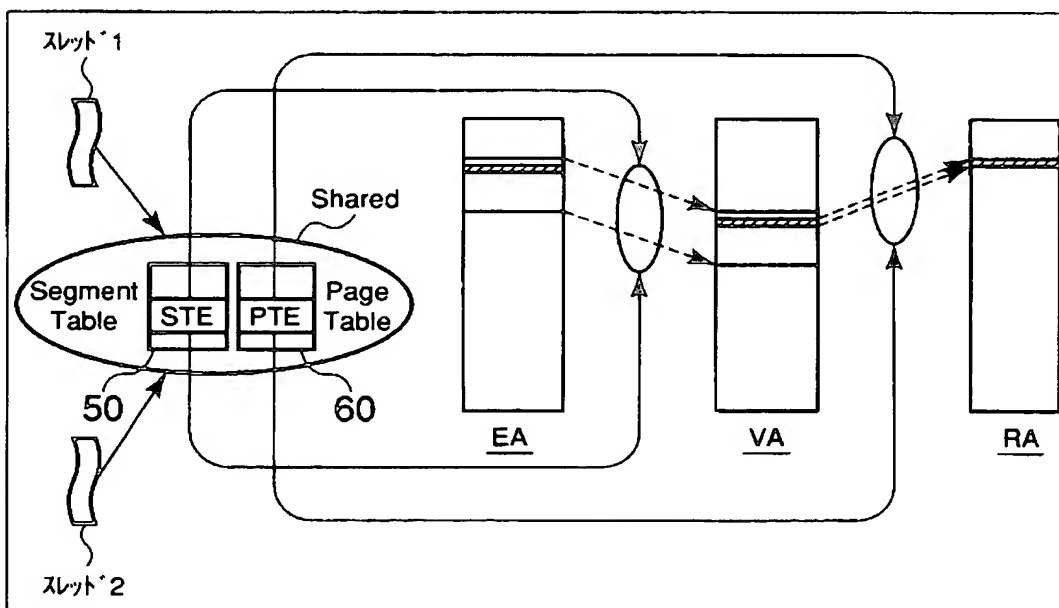
【図 3 3】



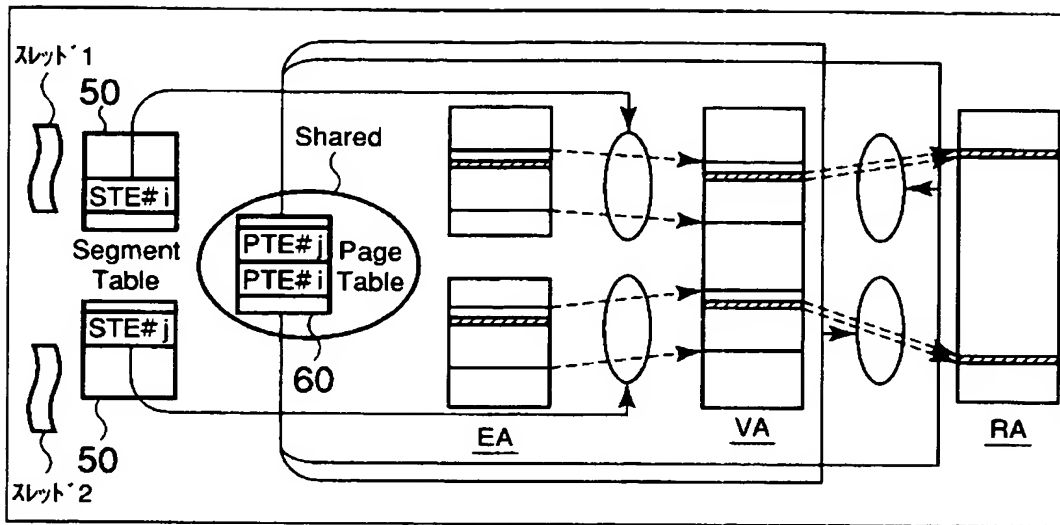
【図 34】



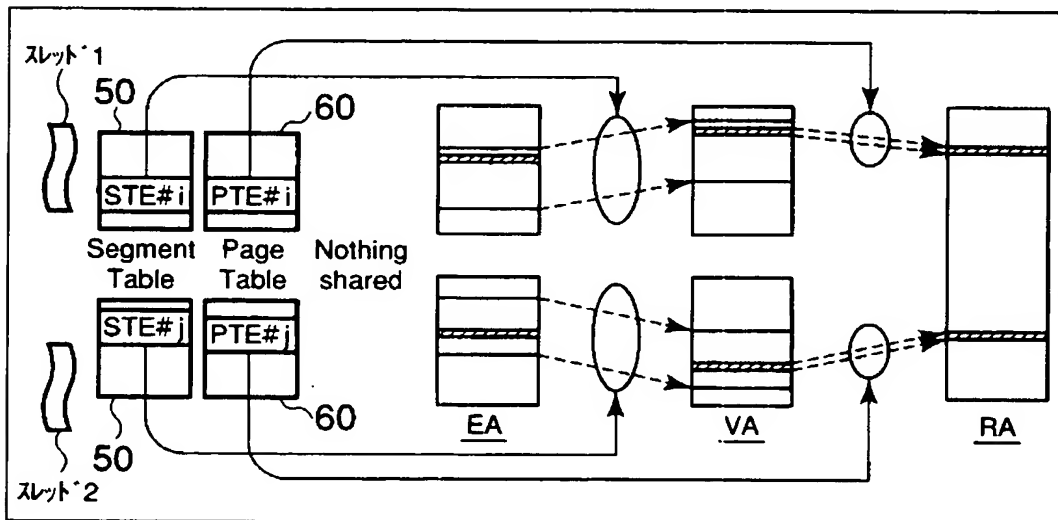
【図 35】



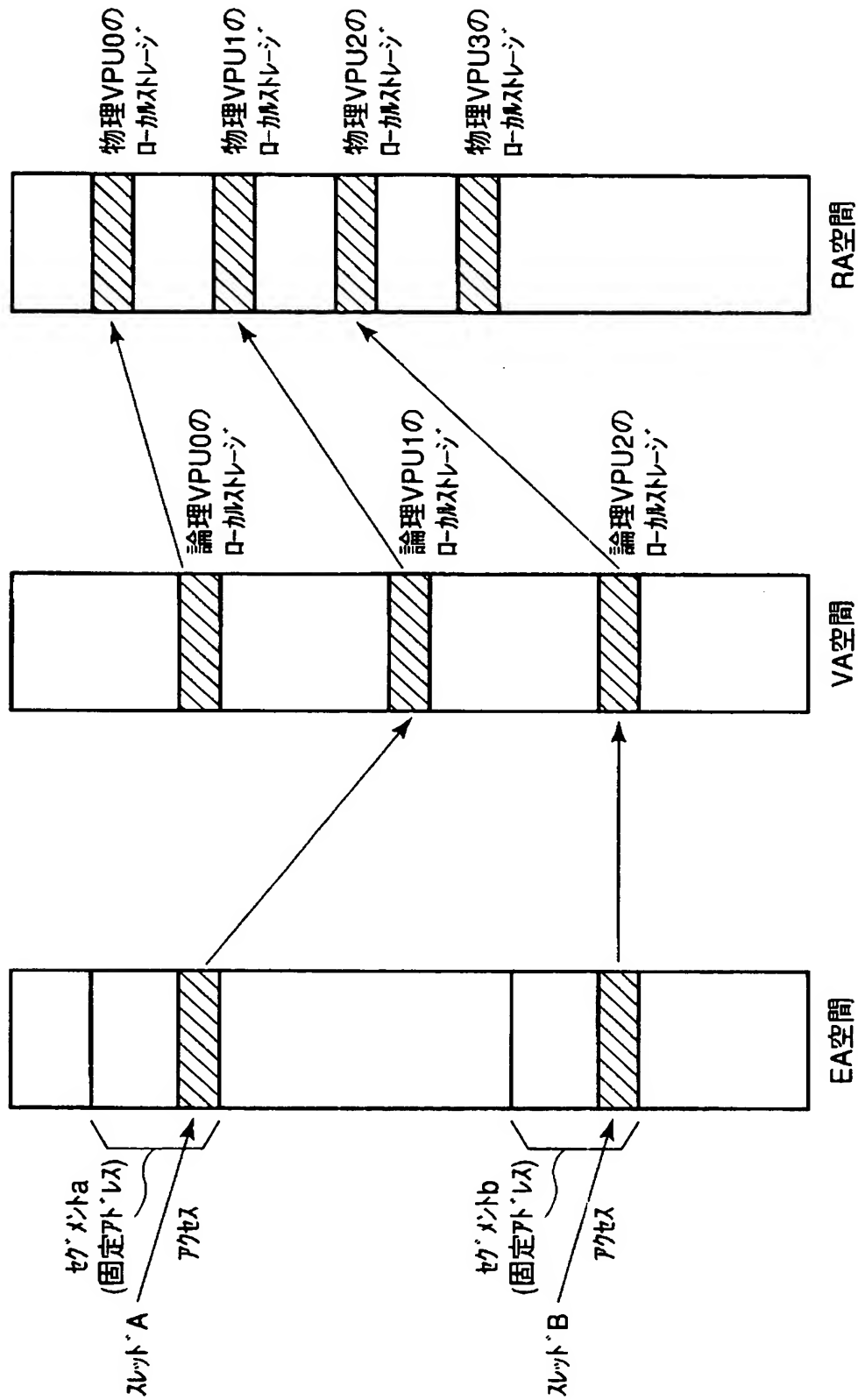
【図 3 6】



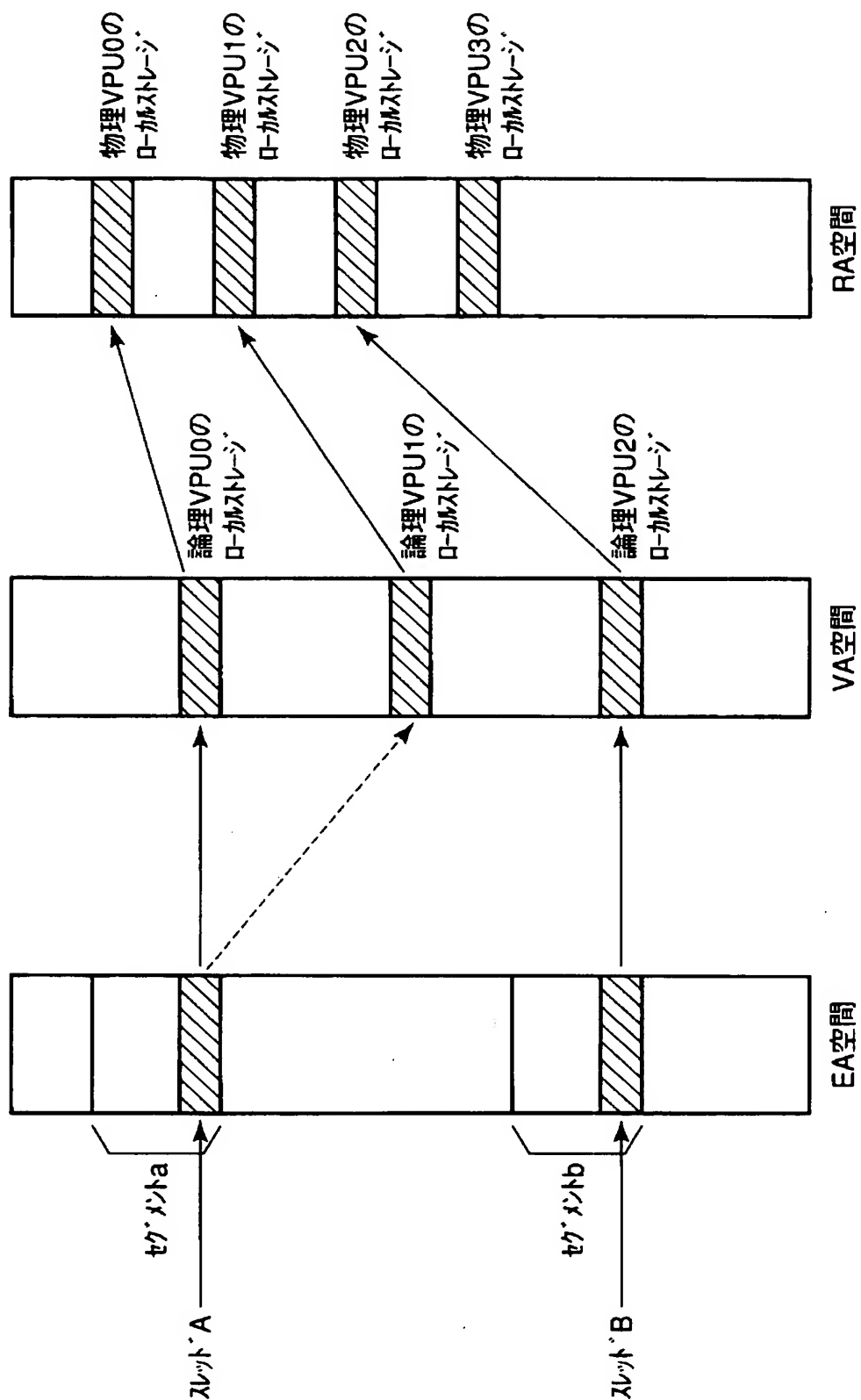
【図 3 7】



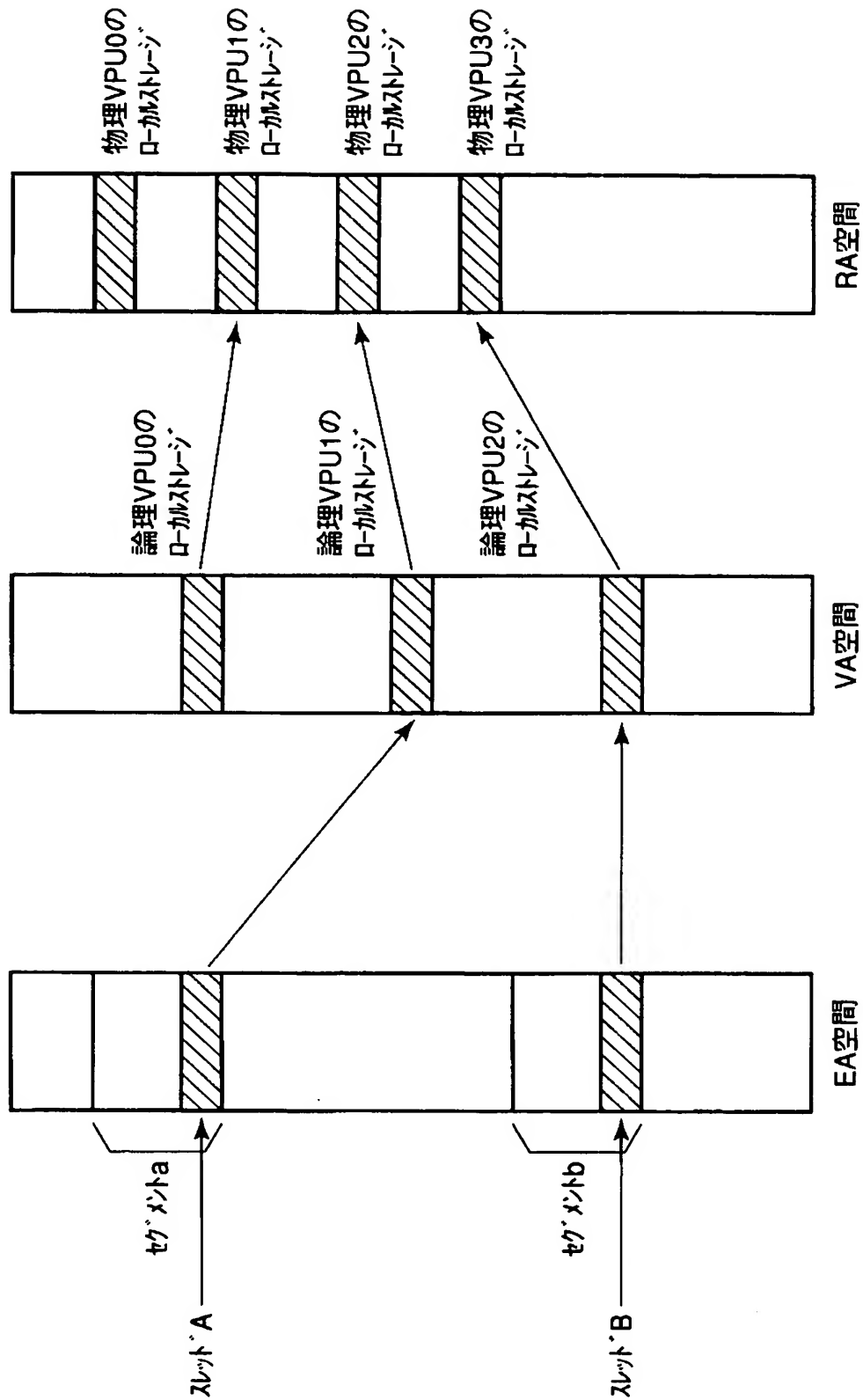
【図 38】



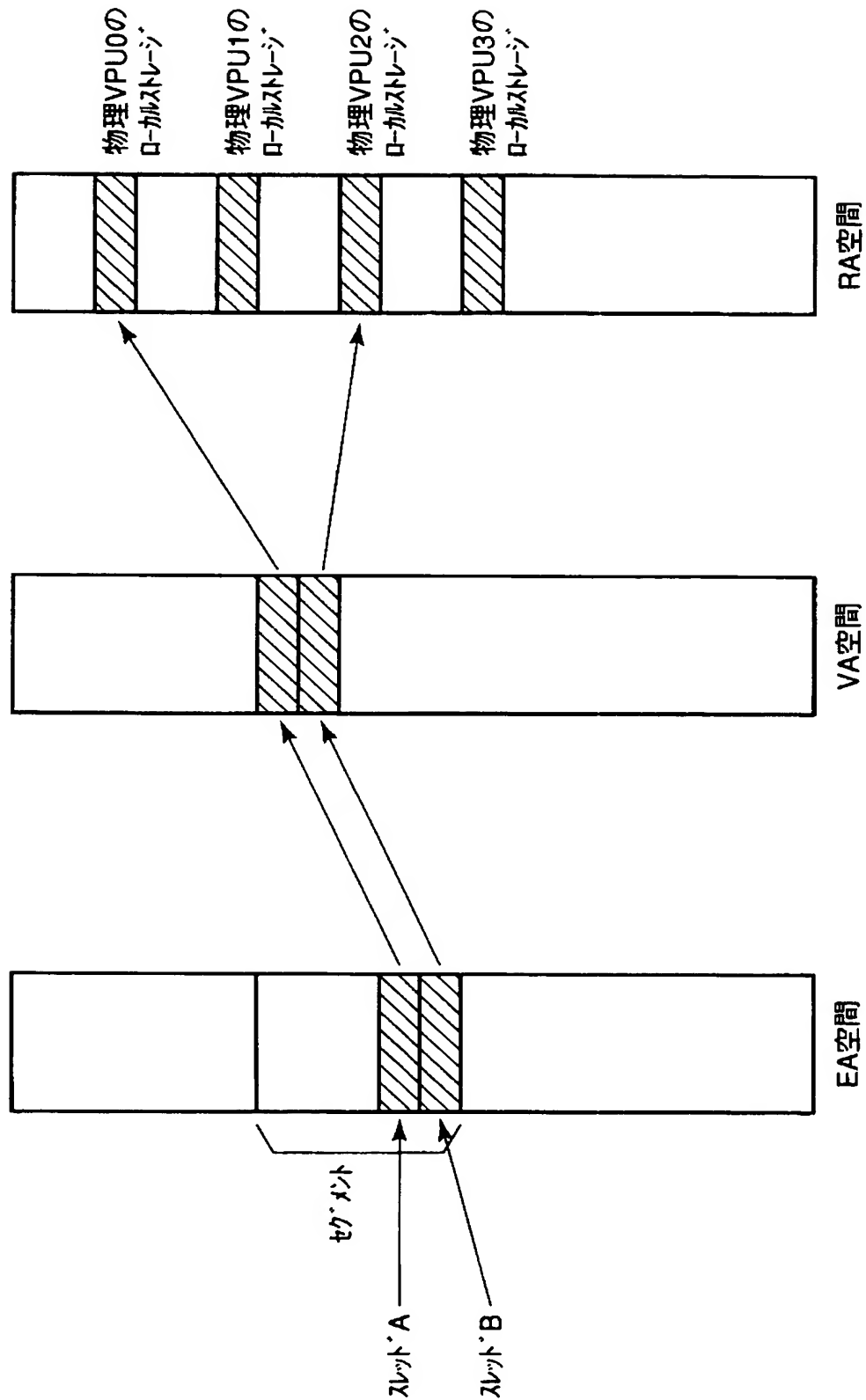
【図 39】



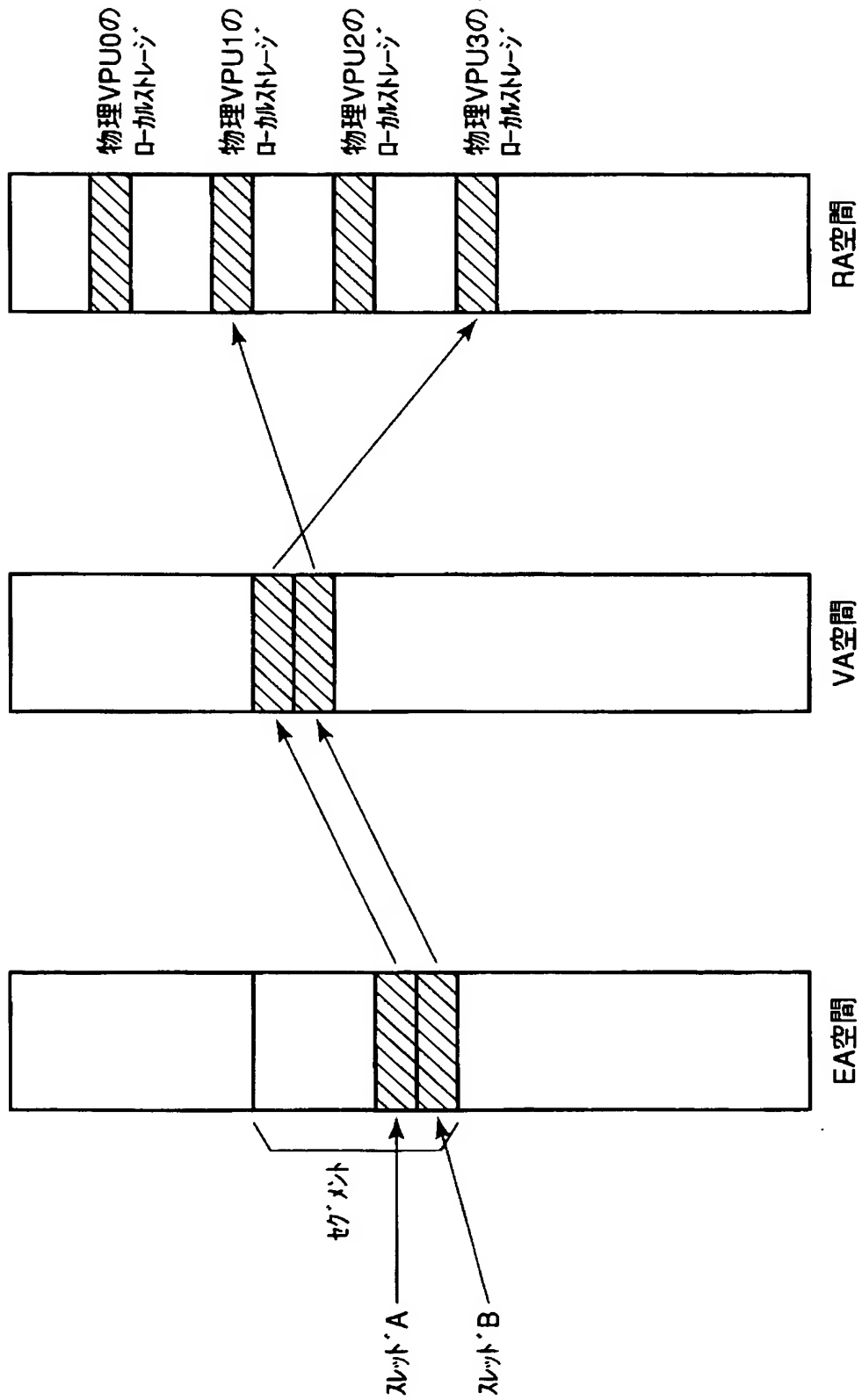
【図 40】



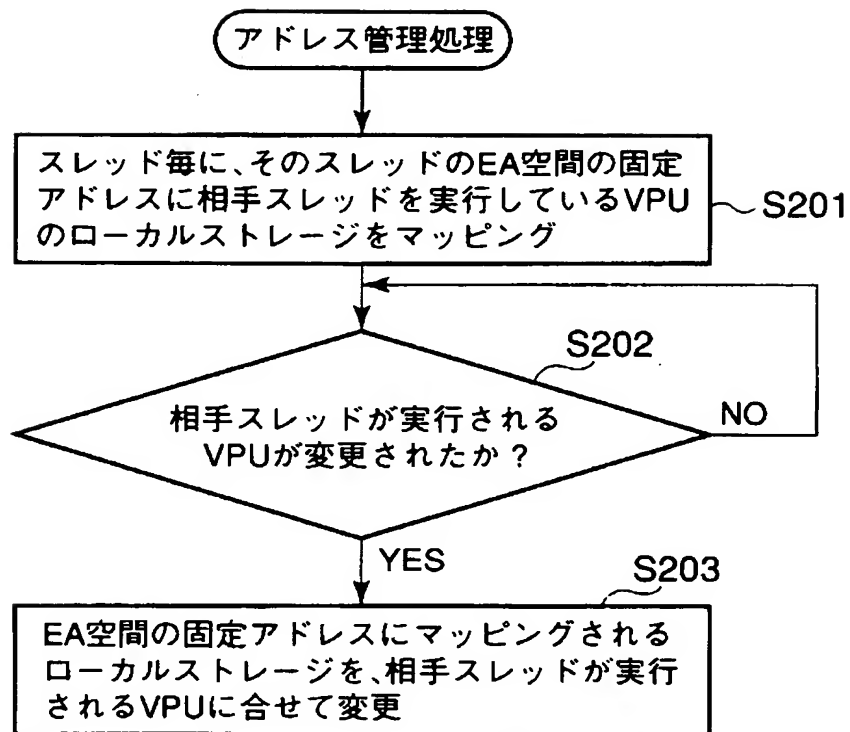
【図 4 1】



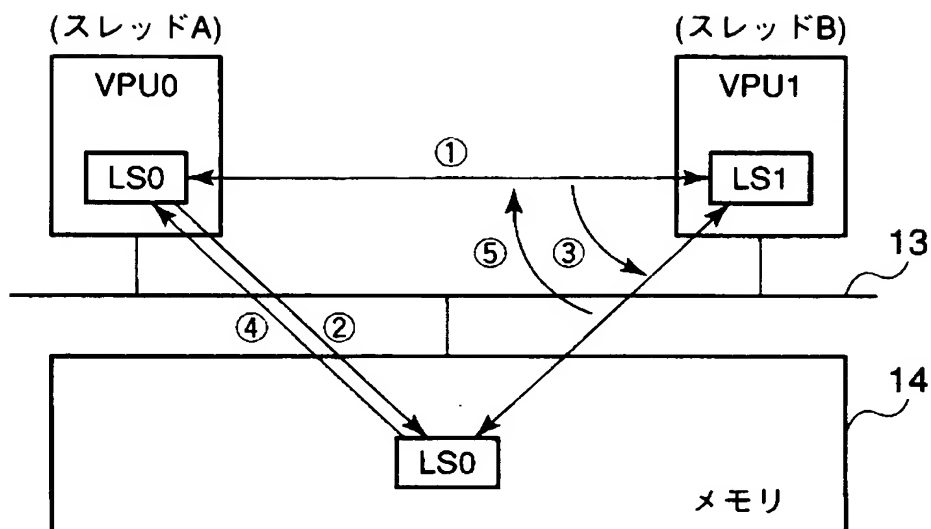
【図 4 2】



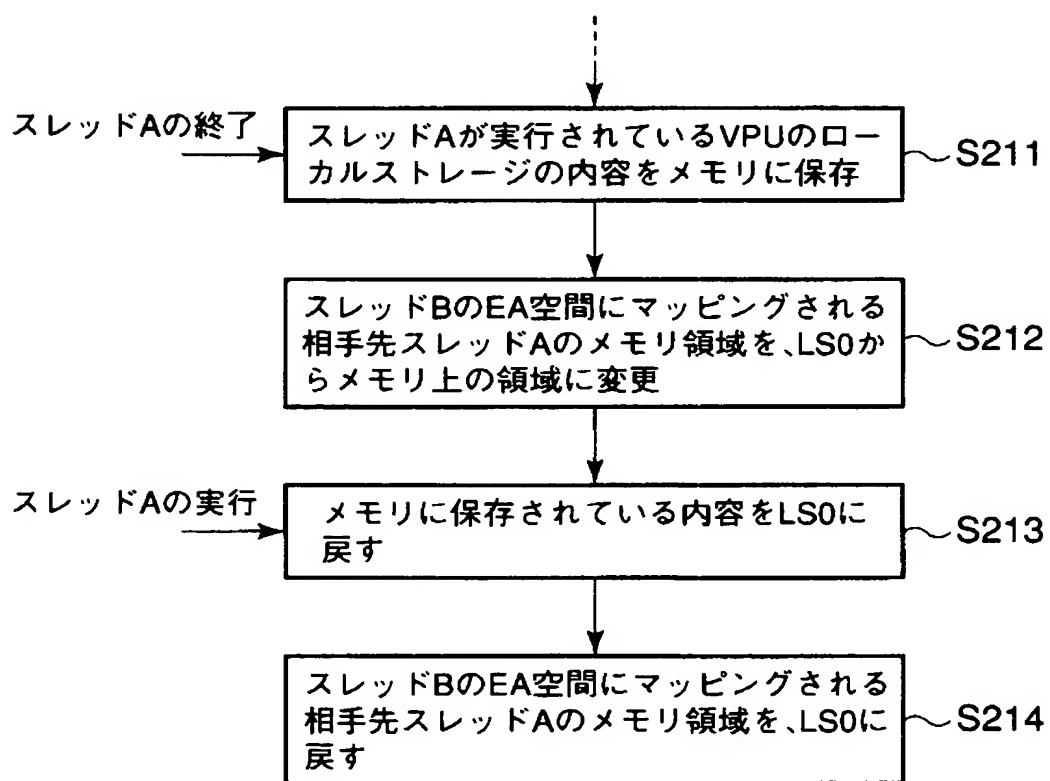
【図 43】



【図 4 4】



【図 4 5】



【書類名】 要約書

【要約】

【課題】複数のプロセッサを用いて複数のスレッドを効率よく並列に実行可能なシステムを実現する。

【解決手段】情報処理システムにおいては、複数のプロセッサLS0, LS1, LS2にローカルメモリLS0, LS1, LS2がそれぞれ設けられているので、各スレッドは共有メモリをアクセスせずとも、プロセッサ内のローカルメモリをアクセスするだけでプログラムを実行することができる。また、相互作用を行う相手のスレッドが実行されるプロセッサに応じて、実効アドレスEA空間にマッピングされる、相手のスレッドに対応するプロセッサのローカルメモリが自動的に変更されるので、各スレッドは相手のスレッドがディスパッチされるプロセッサを意識することなく、相手のスレッドとの相互作用を効率よく行うことが出来る。よって、複数のスレッドを効率よく並列に実行することが可能となる。

【選択図】 図36



特願 2003-185416

出願人履歴情報

識別番号

[000003078]

1. 変更年月日

2001年 7月 2日

[変更理由]

住所変更

住 所

東京都港区芝浦一丁目1番1号

氏 名

株式会社東芝